Apprentissage statistique : théorie et méthodes
M1MINT

Agathe Guilloux

Introduction

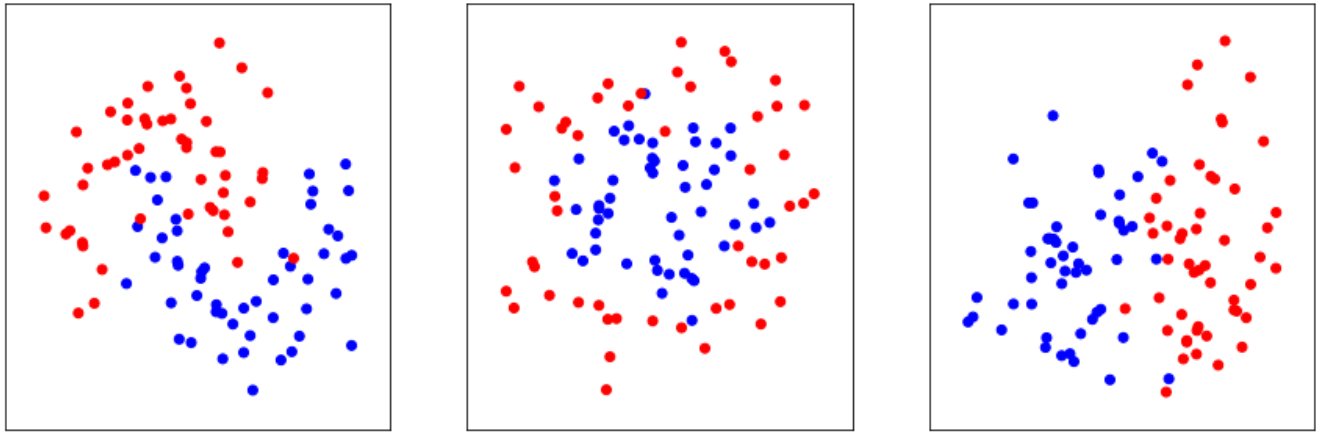
But du cours

- Le but de ce cours est à la fois d'étudier de nouveaux algorithmes de classification mais également d'obtenir des résultats mathématiques sur les algorithmes décrits
- Vous connaissez déjà deux algorithmes de classification (Naive Bayes, trees) et un algorithme de régression (modèle linéaire)
- On parlera beaucoup de classification mais, dans la plupart des cas, cela s'étend aux problèmes d'apprentissage supervisé quand les labels sont continus (voir remarques, exercices)

Spam detection

- Données : emails
- Input : email
- Output : Spam or No Spam

Classification binaire : toy datasets



- But : retrouver la classe
- Input : 2 predicteurs
- Output : classe

Classification multi-classes

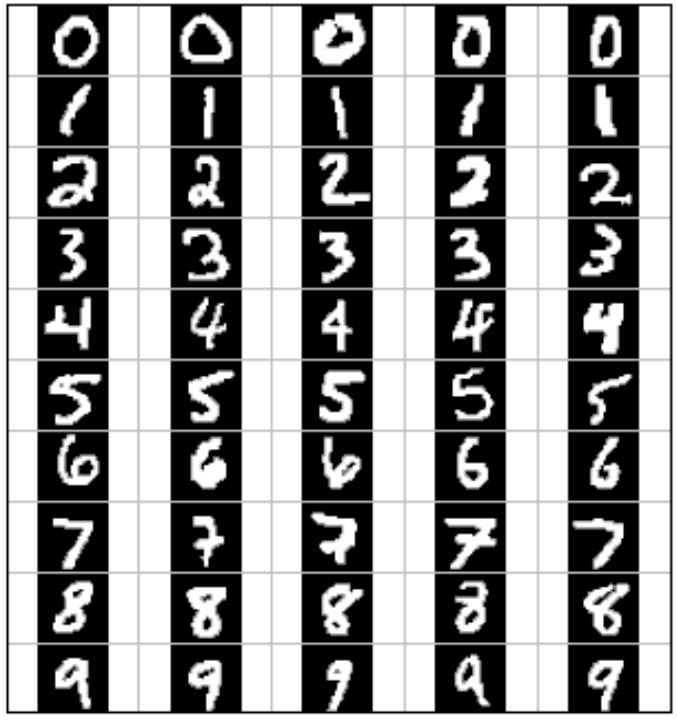
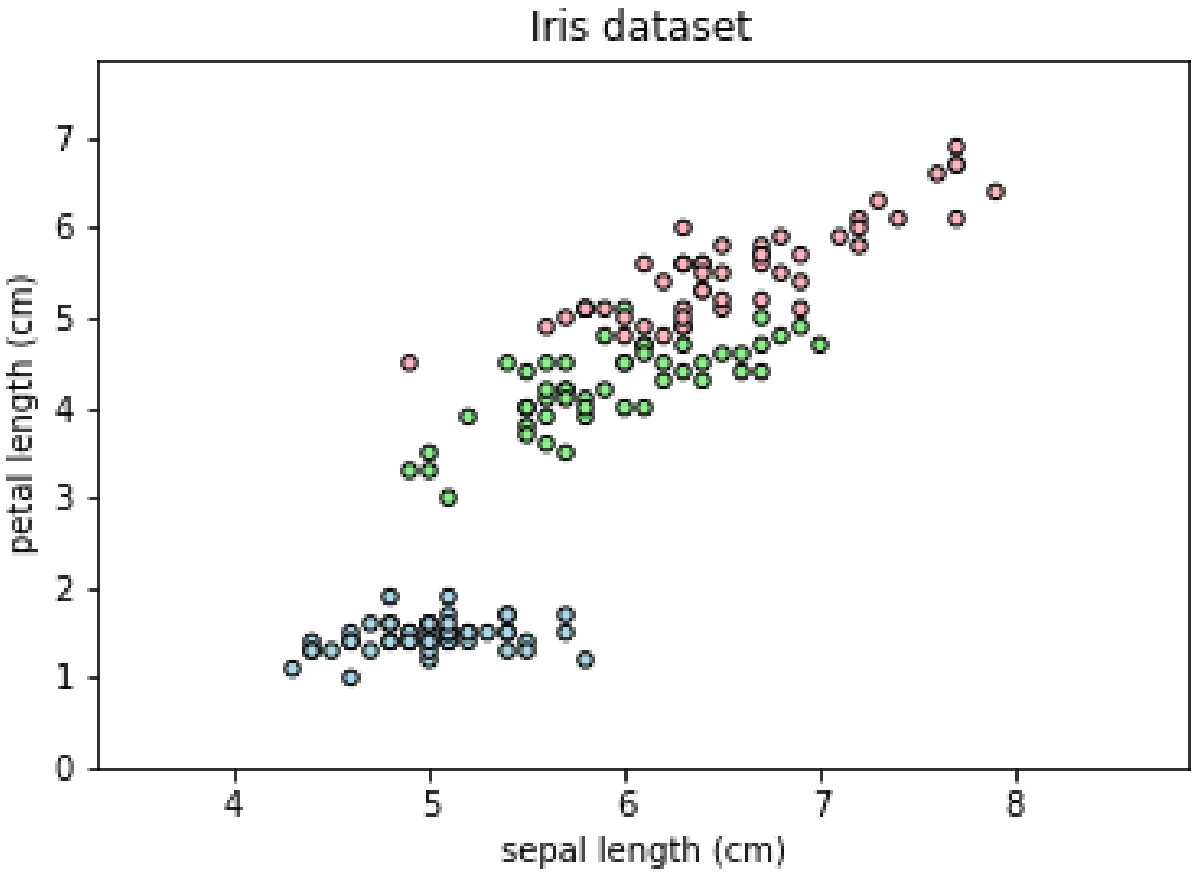


Figure 1: Jeu de données MNIST

- Lire un code postal sur une enveloppe.
- But : assigner un chiffre à une image.
- Input : image.
- Output : chiffre correspondant.

Classification multi-classes : Iris dataset



- But : retrouver la classe
- Input : 2 predicteurs
- Output : classe

Le problème de classification binaire

On a des données d'apprentissage (learning data) pour des individus $i = 1, \ldots, n$. Pour chaque individu i :

- on a un vecteur de covariables (features) $X_i \in \mathcal{X} \subset \mathbb{R}^d$
- la valeur de son label $Y_i \in \{-1, 1\}$.
- on suppose que les couples (X_i, Y_i) sont des copies i.i.d. de (X, Y) de loi inconnue et que l'on observe leurs réalisations (x_i, y_i) $(i = 1, \ldots, n)$.

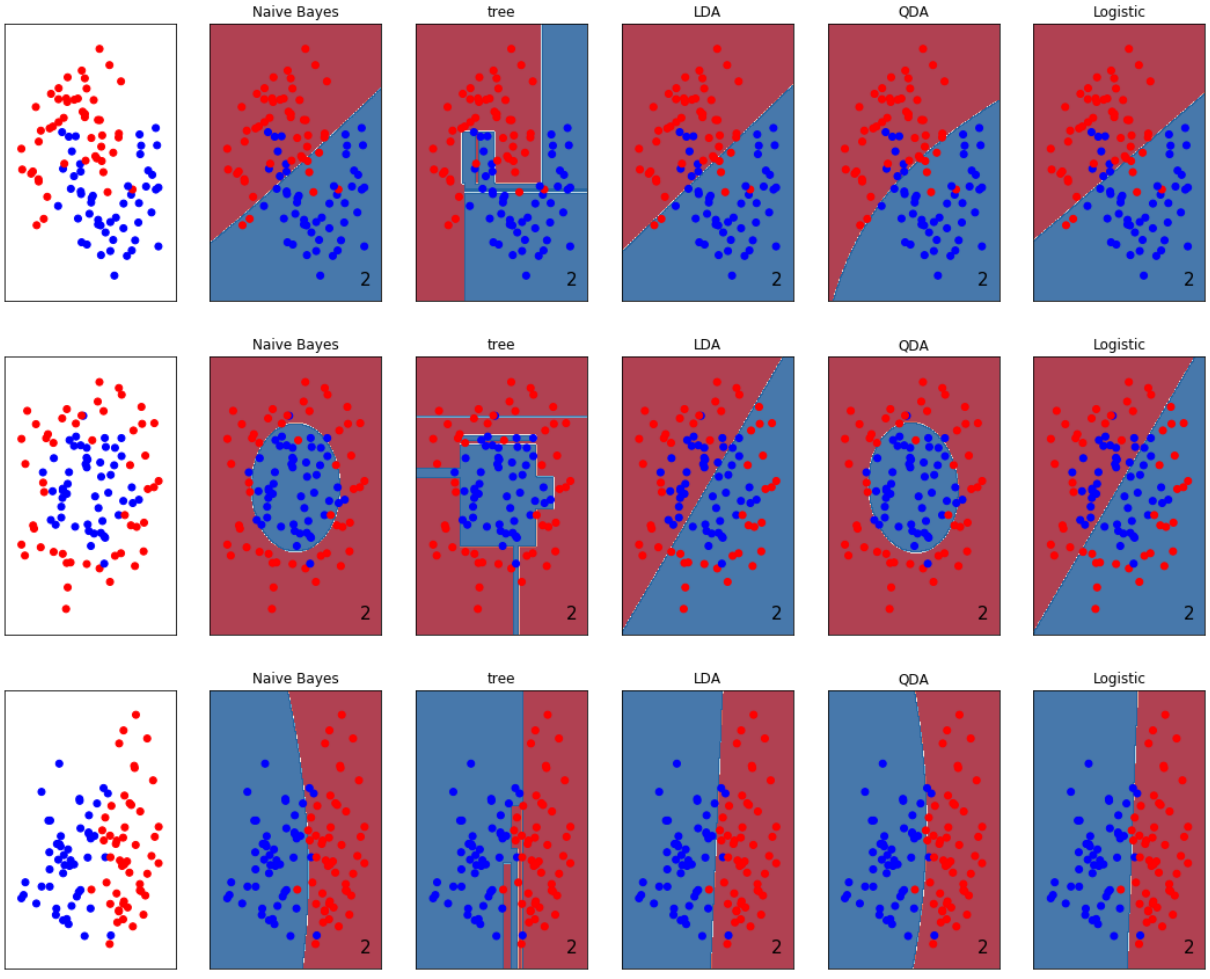
But

- On veut, pour un nouveau vecteur X_+ de features, prédire la valeur du label Y_+ par $\hat{Y}_+ \in \{-1, 1\}$
- Pour cela, on utilise les données d'apprentissage $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ pour construire un **classifieur** $\hat{c}$ de telle sorte que

$$\hat{Y}_+ = \hat{c}(X_+).$$

et $\hat{Y}$ est proche de Y_+ (dans un sens à préciser).

Classification binaire : toy datasets



Le problème de classification multi-classes

On a des données d'apprentissage (learning data) pour des individus $i = 1, \ldots, n$. Pour chaque individu i :

- on a un vecteur de covariables (features) $X_i \in \mathbb{R}^d$
- la valeur de son label $Y_i \in \mathcal{C} = \{1, \ldots, K\}$.
- on suppose que les couples (X_i, Y_i) sont des copies i.i.d. de (X, Y) de loi inconnue et que l'on observe leurs réalisations (x_i, y_i) $(i = 1, \ldots, n)$.

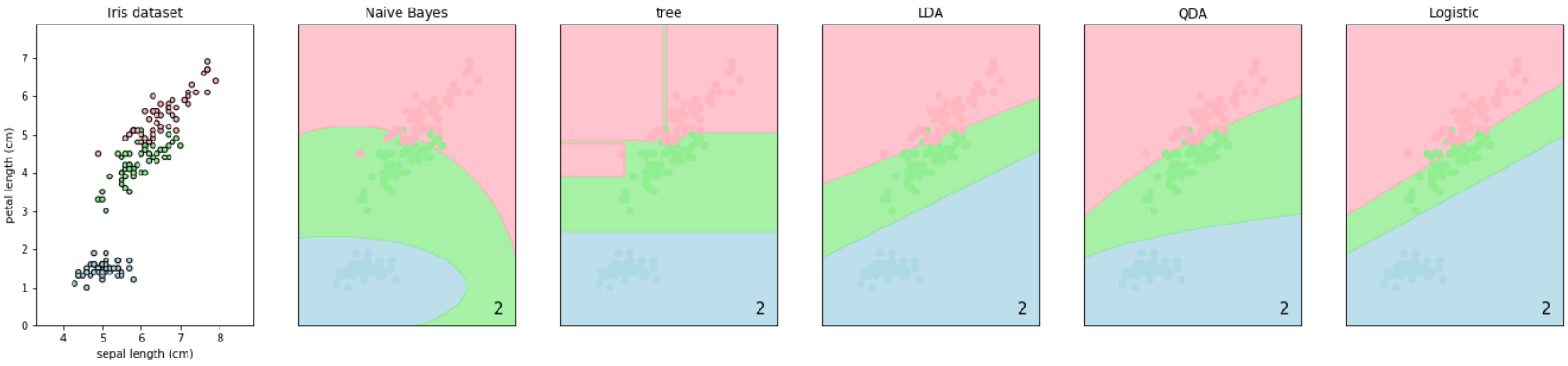
But

- On veut pour un nouveau vecteur X_+ de features prédire la valeur du label Y_+ par $\hat{Y}_+ \in \mathcal{C} = \{1, \ldots, K\}$
- Pour cela, on utilise les données d'apprentissage $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ pour construire un **classifieur** $\hat{c}$ de telle sorte que

$$\hat{Y}_+ = \hat{c}(X_+)$$

et $\hat{Y}_+$ est proche de Y_+ (dans un sens à préciser).

Classification multi-classes : Iris dataset



Organisation du cours

- 10/12 Régression logistique, pénalisations, métriques en classification
- 17/12 Cours/TD 1
- 07/01 Support vector machines et noyaux
- 14/01 TP 1
- 28/02 Boosting, random forests, perceptron
- 04/02 TP 2
- 11/02 Bornes de risques
- 18/02 Cours/TD 2

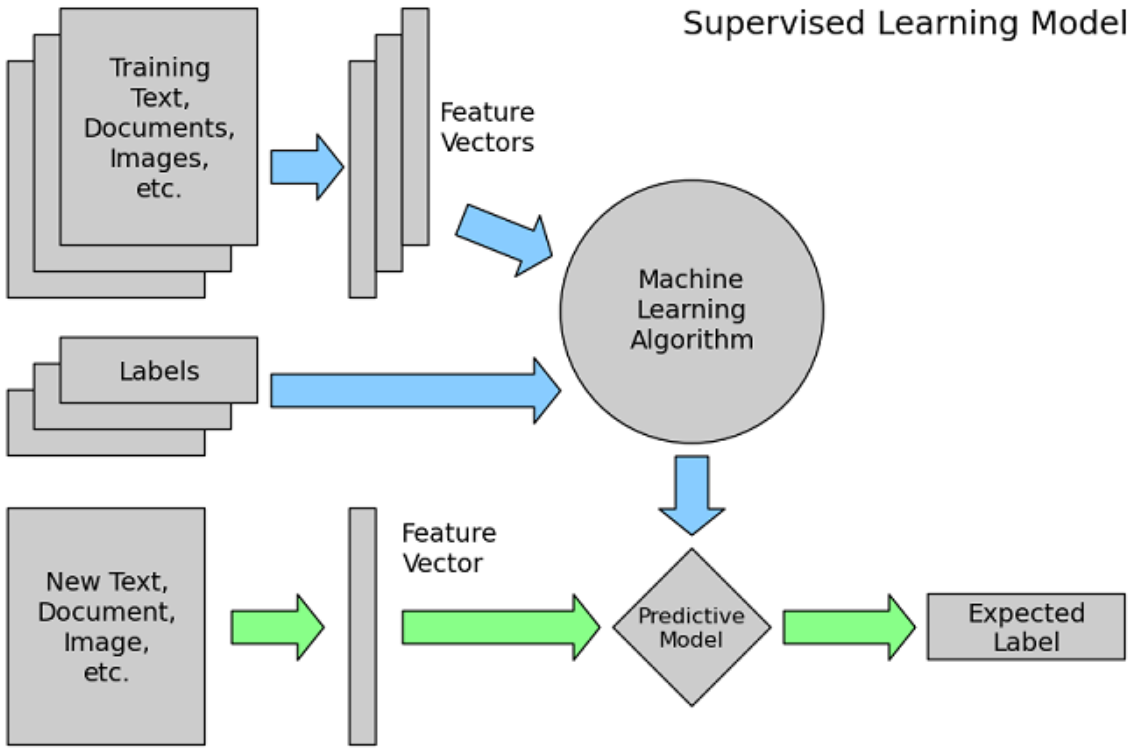
Références

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, 2001.

Max Kuhn and Kjell Johnson. *Applied predictive modeling*. Vol. 810. Springer, 2013.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

Apprentissage statistique supervisé



- Input : covariables, variables explicatives, features $X = (X^1, \ldots, X^d)$
- Ouput : variable à expliquer, variable dépendante, réponse, label Y

Régression logistique

Approche probabiliste / statistique en classification binaire

- Pour construire le classifieur $\hat{c}$, on construit des estimateurs $\hat{p}_1(x)$ et $\hat{p}_{-1}(x)$ de

$$p_1(x) = \mathbb{P}(Y = 1 | X = x) \quad \text{et} \quad p_{-1}(x) = 1 - p_1(x)$$

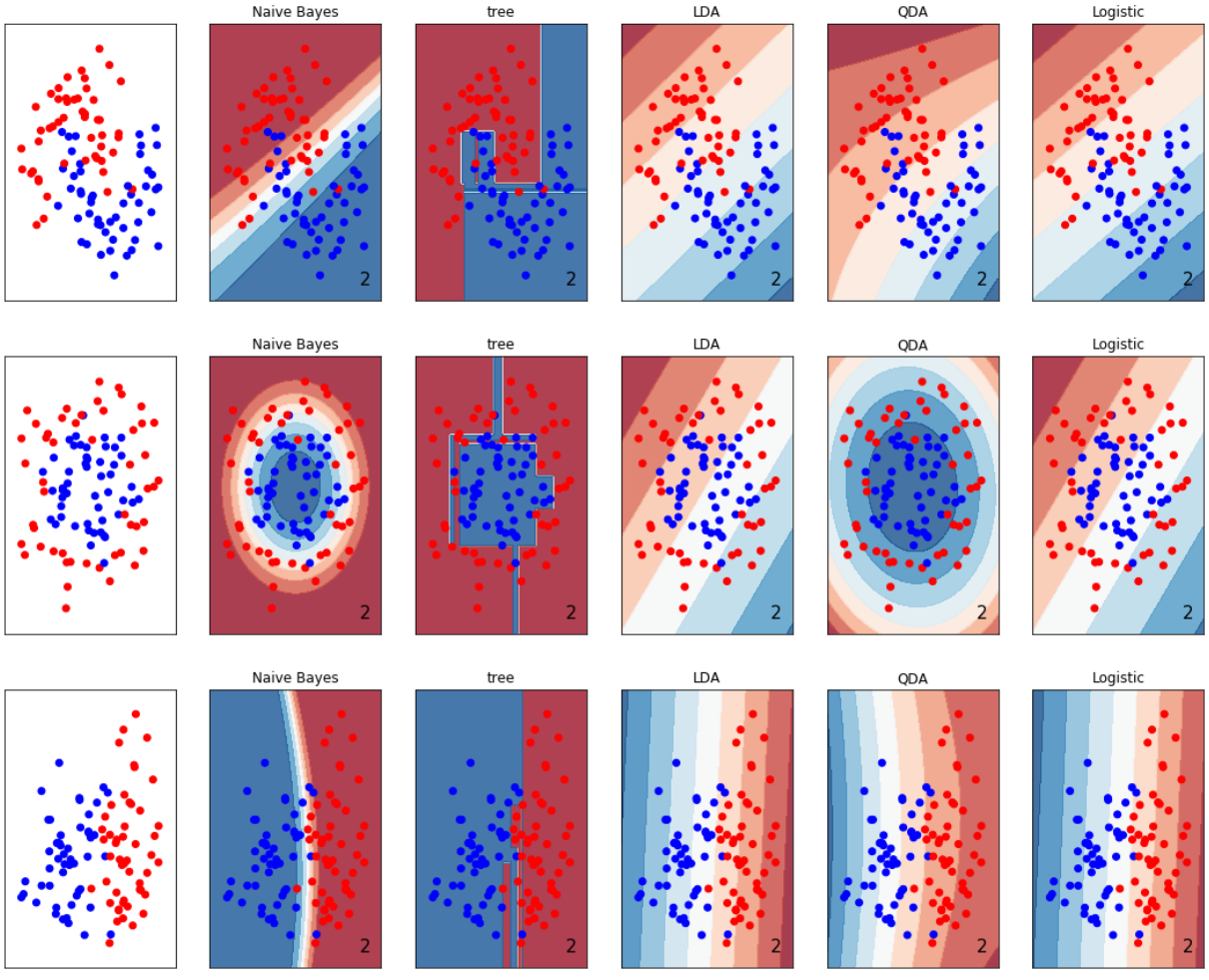
- en modélisant la loi de $Y|X$.
- Puis, conditionnellement à $X_+ = x$, on classifie en utilisant la règle

$$\hat{Y}_+ = \hat{c}(x) = \begin{cases} 1 & \text{si } \hat{p}_1(x) \geq s \\ -1 & \text{sinon} \end{cases}$$

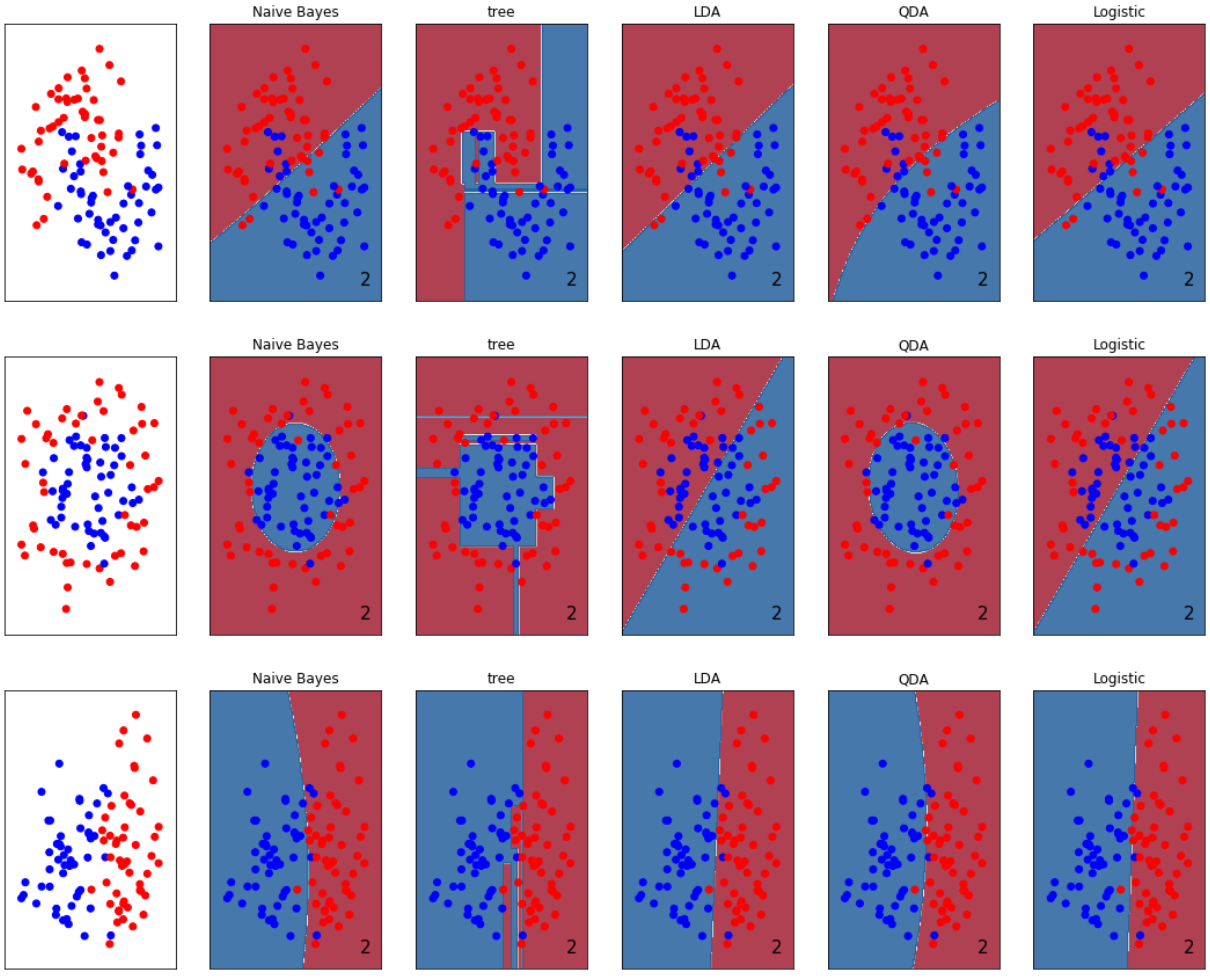
pour un seuil $s \in (0, 1)$.

- Si on choisit $s = 1/2$, cela revient à classifier suivant la plus grande valeur entre $\hat{p}_1(x)$ et $\hat{p}_{-1}(x)$ (on retient cette règle dans la suite).

Classification binaire : toy datasets



Classification binaire : toy datasets



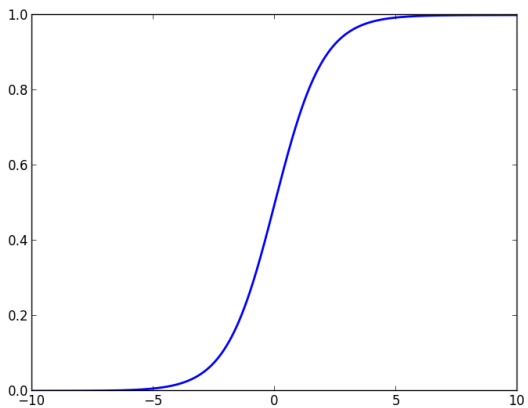
Régression logistique

- C'est le plus utilisé des algorithmes de classification
- On modélise la loi de $Y|X$ par

$$\mathbb{P}(Y = 1|X = x) = \sigma(\langle x, w\rangle + b)$$

où $w \in \mathbb{R}^d$ est un vecteur de régression ou de poids, $b \in \mathbb{R}$ est the **intercept**, et σ est la fonction **sigmoïde**

$$\sigma(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}$$



Autres régressions

- Le choix de la fonction sigmoïde est lié à la loi de Bernoulli.
- On peut considérer d'autres fonctions de $\mathbb{R} \to [0,1]$ (car on veut modéliser une proba). Par exemple, toutes les fonctions de répartition

$$\mathbb{P}(Y = 1|X = x) = F(\langle x, w\rangle + b).$$

- Parmi celles-ci, la f.d.r. gaussienne est souvent utilisée

$$F(z) = \Phi(z) = \mathbb{P}(N(0,1) \leq z),$$

et on parle alors de régression **probit**.

- En classification multi-classes, on utilise la fonction **softmax** donnée par

$$\mathbb{P}(Y = k|X = x) = \frac{\exp(\langle x, w_k\rangle + b_k)}{\sum_{k=1}^{K} \exp(\langle x, w_k\rangle + b_k)}$$

pour tout $k \in \mathcal{C} = \{1, \ldots, K\}$.

Rapport de côtes ou odd-ratios

Définition : odds ou côte

La quantité $\frac{\mathbb{P}(Y=1|X=x)}{\mathbb{P}(Y=-1|X=x)}$ est appelée **odds** ou côte.

Dans le modèle logistique, on a défini l'odds (ou la côte) par

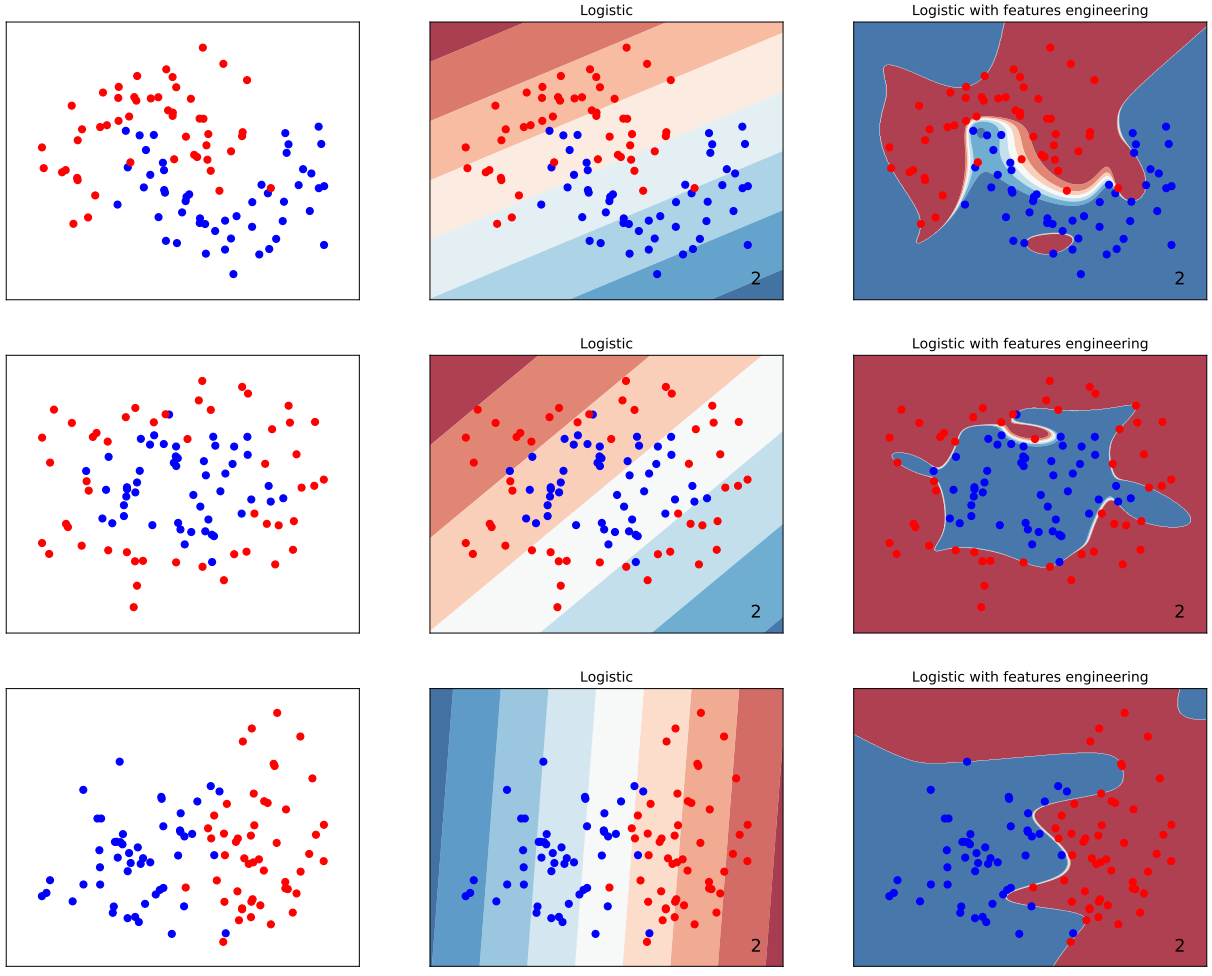
$$\frac{\mathbb{P}(Y=1|X=x)}{\mathbb{P}(Y=-1|X=x)} = \exp(\langle x, w\rangle + b) = \exp(b + w_1 x^1 + \ldots + w_p x^p).$$

On considère deux individus i_1 et i_2 dont la valeur des covariables ne diffère que pour la j-ième covariable avec $x^j_{i_1} - x^j_{i_2} = 1$, on calcule l'odds-ratio (ou le rapport des côtes)

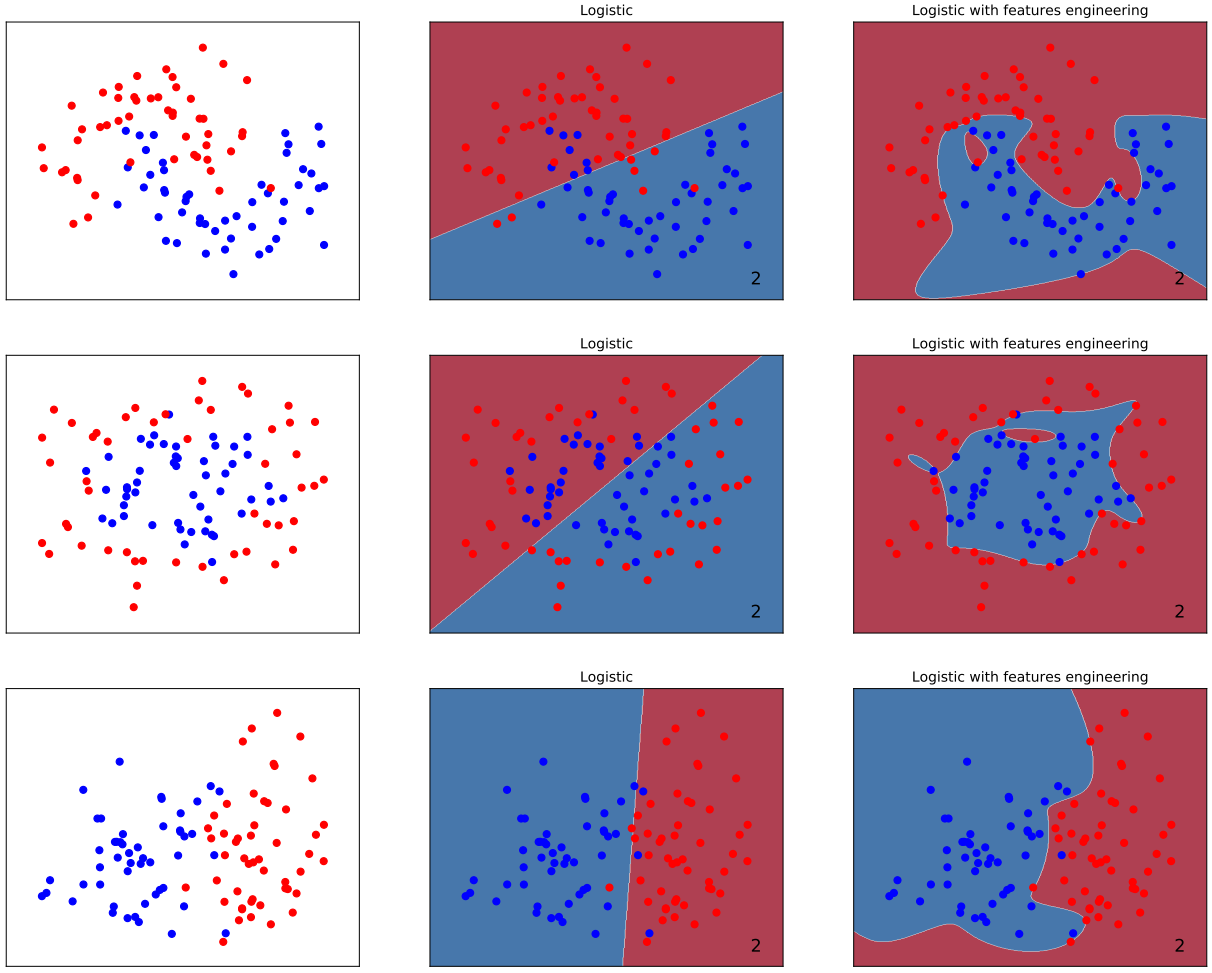
$$\frac{\mathbb{P}(Y=1|X=x_{i_1})}{\mathbb{P}(Y=-1|X=x_{i_1})} \Big/ \frac{\mathbb{P}(Y=1|X=x_{i_2})}{\mathbb{P}(Y=-1|X=x_{i_2})} = \exp(w_j)$$

On dira alors qu'une augmentation de 1 de la variable j entraîne une multiplication de l'odds de $\exp(w_j)$.

Illustration



Illustration



Règle de classification linéaire

- Remarquons que

$$\mathbb{P}(Y = 1|X = x) \geq \mathbb{P}(Y = -1|X = x)$$

  si et seulement si

$$\langle x, w \rangle + b \geq 0.$$

- On obtient une règle de classification linéaire, c'est-à-dire linéaire par rapport aux features
- Mais, on peut faire du **features engineering** (considérer comme covariables x^1,x^2, leur produit et leurs carrés, etc)

Estimation de w et b

- Nous avons un modèle pour la loi de $Y|X$
- sous l'hypothèse (X_i, Y_i) i.i.d.
- on calcule des estimateurs $\hat{w}$ et $\hat{b}$ via **le maximum de vraisemblance**
- ou, de façon équivalente, on minimise $1/n$ la log-vraisemblance négative
- Dans ce cas, on dira

$$\text{Empirical loss} = \text{Goodness-of-fit} = \text{Train error} = -\frac{1}{n}\log \text{ likelihood}$$

Exercice sur la régression linéaire

Perte empirique

On considère maintenant que $Y \in \mathbb{R}$ et que la loi de $Y|X = x$ est la loi $\mathcal{N}(\langle x, w \rangle + b, \sigma^2)$ Quelle est la perte empirique dans ce modèle linéaire ?

Calcul des estimateurs (1)

La vraisemblance est donnée par

$$\prod_{i=1}^{n} \mathbb{P}(Y = y_i | X = x_i)$$
$$= \prod_{i=1}^{n} \sigma(\langle x_i, w \rangle + b)^{\frac{y_i}{2}} \left(1 - \sigma(\langle x_i, w \rangle + b)\right)^{\frac{1-y_i}{2}}$$

et $-1/n$ la log-vraisemblance par

$$\sum_{i=1}^{n} \log(1 + e^{-y_i(\langle x_i, w \rangle + b)})$$

Calcul des estimateurs (2)

$\hat{w}$ et $\hat{b}$ vérifient :

$$(\hat{w}, \hat{b}) \in \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_i(\langle x_i, w \rangle + b)})$$

- C'est un problème d'optimisation convexe et différentiable
- qu'on peut résoudre par descente de gradient, Newton, etc

Si on définit la fonction (individuelle) de **perte logistique**

$$\ell(y, y') = \log(1 + e^{-yy'})$$

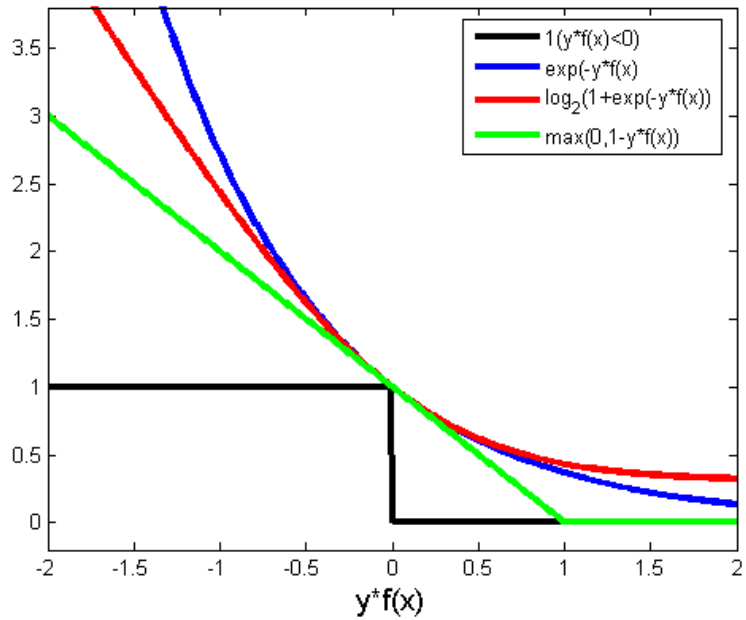
alors

$$\text{Goodness-of-fit} = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b),$$

c'est donc **une perte moyenne** sur les données d'apprentissage.

Autres pertes classiques pour la classification binaire

- Hinge loss (SVM), $\ell(y, y') = (1 - yy')_+$
- Quadratic hinge loss (SVM), $\ell(y, y') = \frac{1}{2}(1 - yy')_+^2$
- Huber loss $\ell(y, y') = -4yy' \mathbb{1}_{yy' < -1} + (1 - yy')_+^2 \mathbb{1}_{yy' \geq -1}$



- Toutes ces pertes peuvent être vus comme des approximations convexes de la perte 0/1 $\ell(y, y') = \mathbb{1}_{yy' \leq 0}$

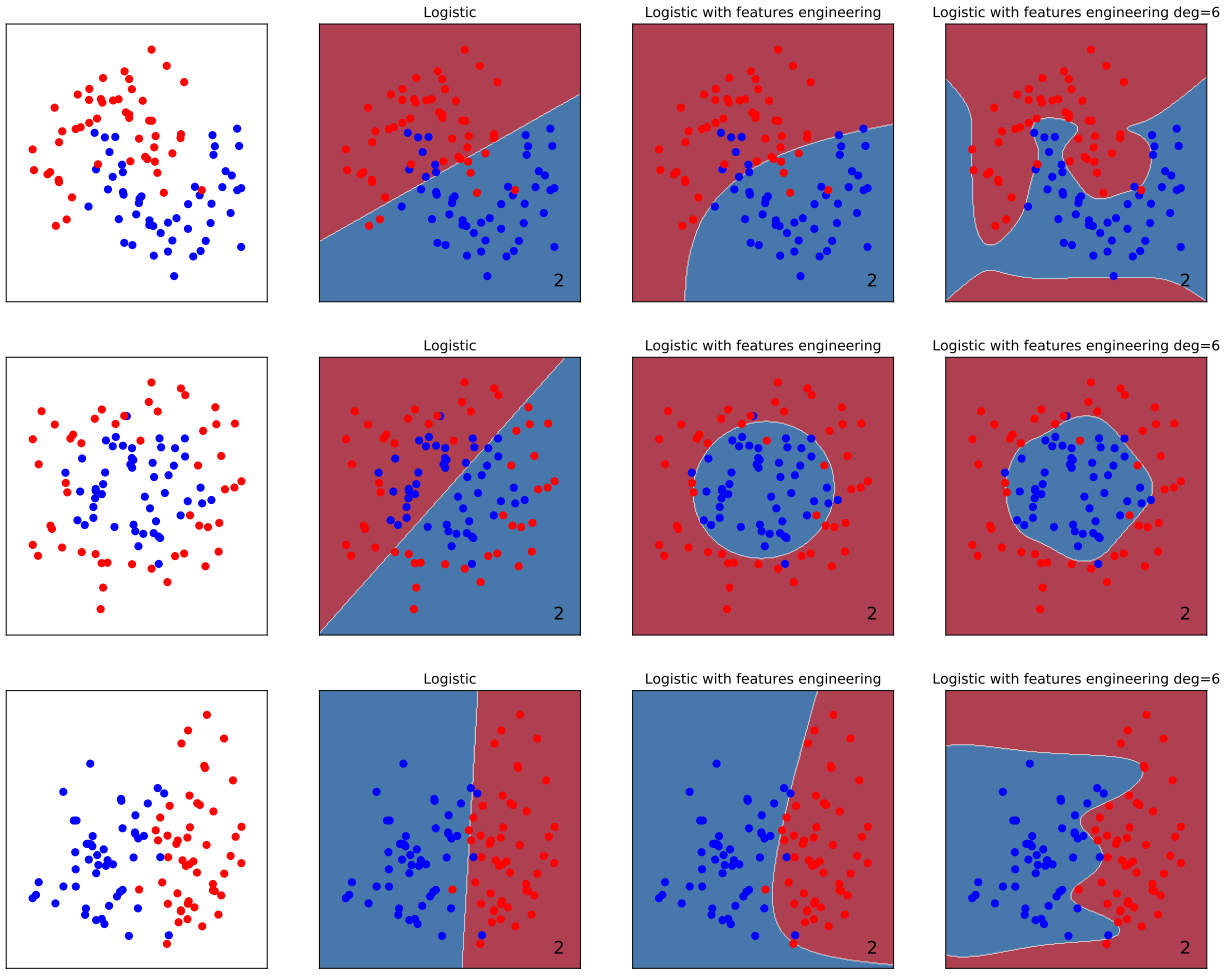
Exercice sur la régression linéaire

Fonction de perte

On considère maintenant que $Y \in \mathbb{R}$ et que la loi de $Y|X = x$ est la loi $\mathcal{N}(\langle x, w \rangle + b, \sigma^2)$ Quelle est la fonction de perte dans ce modèle linéaire ? On l'appellera la perte des moindres carrés, **least squared loss**.

Pénalisations

Exemple


Logistic
Logistic with features engineering
Logistic with features engineering deg=6
Logistic
Logistic with features engineering
Logistic with features engineering deg=6
Logistic
Logistic with features engineering
Logistic with features engineering deg=6

Pénalisation

En définissant

$$\hat{w}, \hat{b} \in \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b)$$

on définit en général un mauvais classifieur, notamment dans les cas où il y a beaucoup de features.

On considère plutôt

$$\hat{w}, \hat{b} \in \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b) + \frac{1}{C} \operatorname{pen}(w) \right\}$$

où

- pen est un terme de **pénalisation**, ça permettra à w ne pas pas être trop "complexe"
- $C > 0$ est un paramètre qui contrôle la force de la pénalisation (appelé paramètre de **tuning** ou de **smoothing**)

Pénalisation ridge

La pénalisation **ridge** est définie par

$$\text{pen}(w) = \frac{1}{2}\|w\|_2^2 = \frac{1}{2}\sum_{j=1}^{d} w_j^2$$

Elle pénalise la taille de w.

- C'est simple
- Elle permet de "régler" les problèmes de corrélation entre variables
- Elle aide par ailleurs les algorithmes d'optimisation (problème plus simple)

Interprétation géométrique

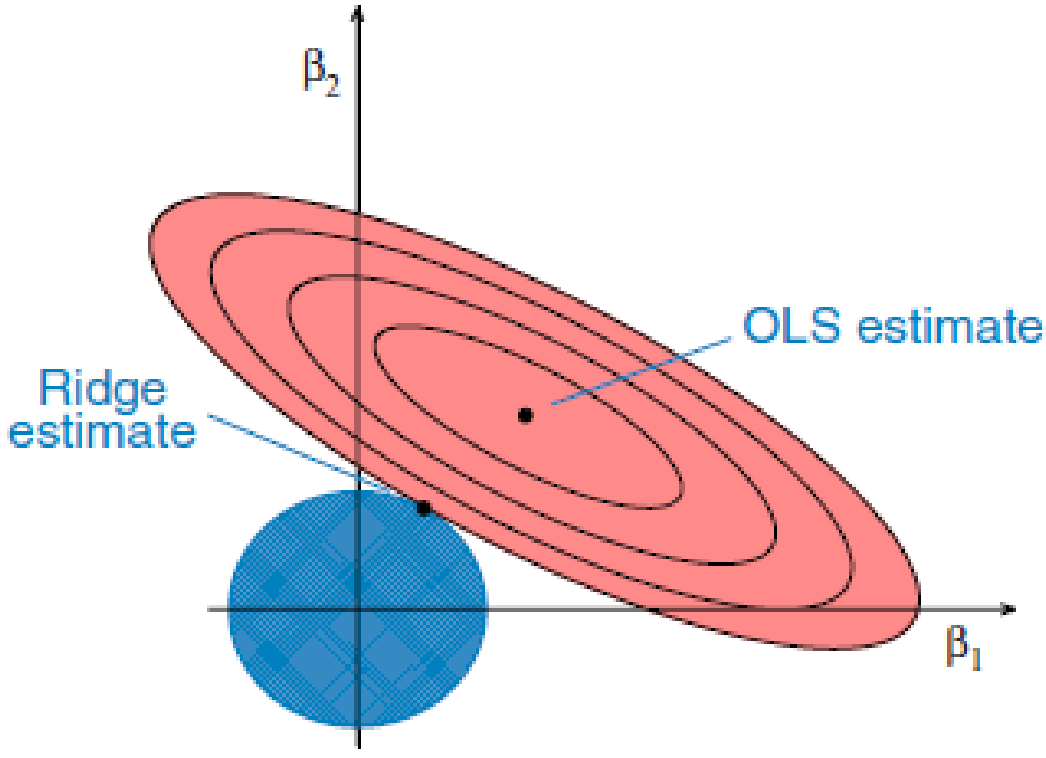


Figure 2: from https://online.stat.psu.edu/stat508/

Sparsité

On remarque que, si $\hat{w}_j = 0$, alors la feature j n'a pas d'impact sur la prédiction

$$\hat{y} = \text{sign}(\langle x, \hat{w} \rangle + \hat{b})$$

Si on a beaucoup de features (si d est grand), on aimerait obtenir un $\hat{w}$ qui contient beaucoup de **zeros**.

On obtiendra alors un modèle plus simple avec une dimension "réduite" et donc plus facilement interprétable

Comment faire ?

Pénalisation par la norme $\|\cdot\|_0$

On aimerait définir

$$\hat{w}, \hat{b} \in \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b) + \frac{1}{C} \|w\|_0 \right\},$$

où

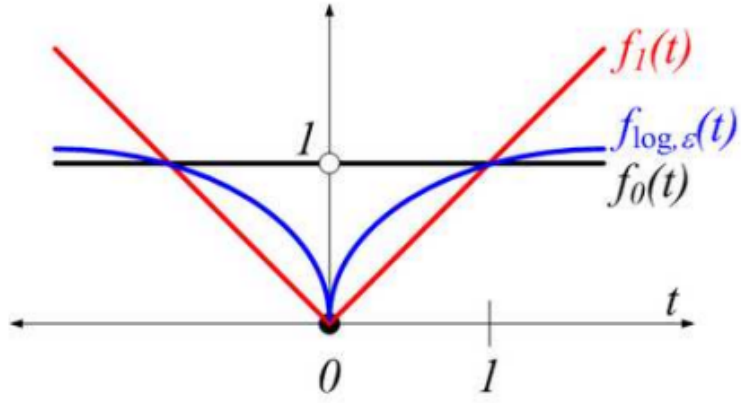
$$\|w\|_0 = \#\{j \in \{1, \dots, d\} : w_j \neq 0\}.$$

Mais, pour résoudre le problème de minimisation qui n'est pas convexe, il faudrait explorer tous les supports possibles de w : c'est trop long (NP-hard)

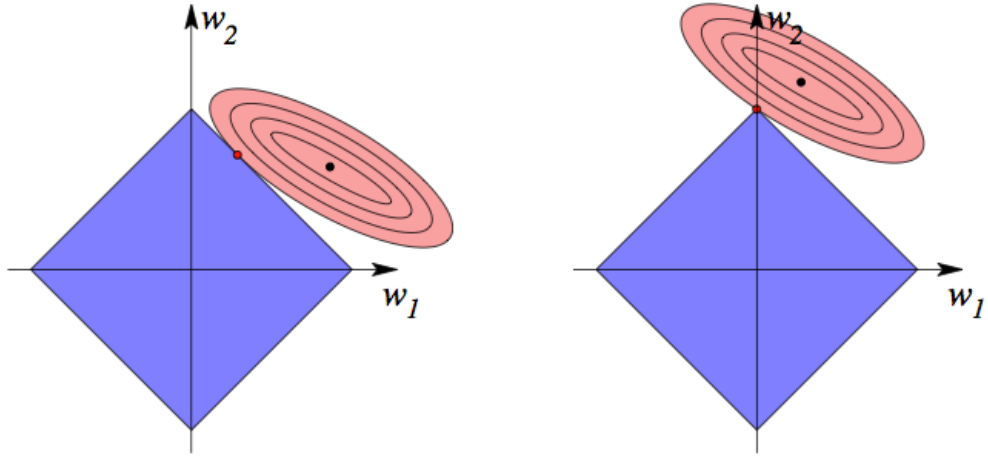
Pénalisation par la norme $\|\cdot\|_1$: le LASSO

Une solution est donc de trouver un "proxy" convexe de la $\|\cdot\|_0$: la **norme** ℓ_1

$\|w\|_1 = \sum_{j=1}^{d} |w_j|$



Pourquoi cela induit-il de la sparsité ?



LASSO vs ridge : interprétation géométrique

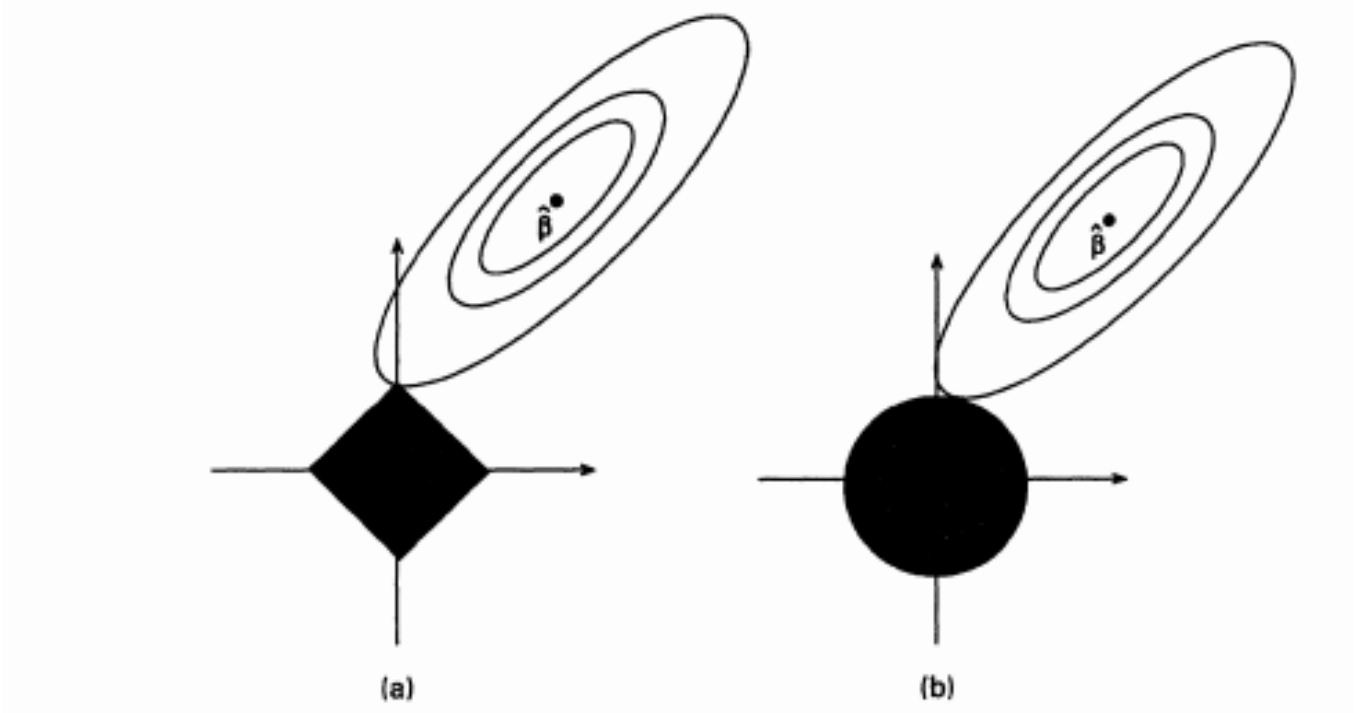


Fig. 2. Estimation picture for (a) the lasso and (b) ridge regression

Régression pénalisée

Considérons le problème de minimisation

$$\hat{w}, \hat{b} \in \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b) + \frac{1}{C} \operatorname{pen}(w) \right\},$$

Pour $\ell(y, y') = \frac{1}{2}(y - y')^2$ et $\operatorname{pen}(w) = \frac{1}{2}\|w\|_2^2$, c'est la **régression ridge**

Pour $\ell(y, y') = \frac{1}{2}(y - y')^2$ et $\operatorname{pen}(w) = \|w\|_1$, c'est le **Lasso** (Least absolute shrinkage and selection operator)

Pour $\ell(y, y') = \log(1 + e^{-yu'})$ et $\operatorname{pen}(w) = \|w\|_1$, c'est la **régression logistique pénalisée** ℓ_1

Il y a de nombreuses combinaisons possibles

Elastic-net

Les combinaisons

$$\text{(régression linéaire ou logistique)} + \text{(ridge or } \ell_1\text{)}$$

sont les plus utilisées.

Une autre pénalité très utilisée est

$$\text{pen}(w) = \frac{1-\alpha}{2}\|w\|_2^2 + \alpha\|w\|_1$$

appelée **elastic-net**, elle bénéfie des avantages des pénalisations ridge et ℓ_1 ($\alpha \geq 0$ équilibre les deux)

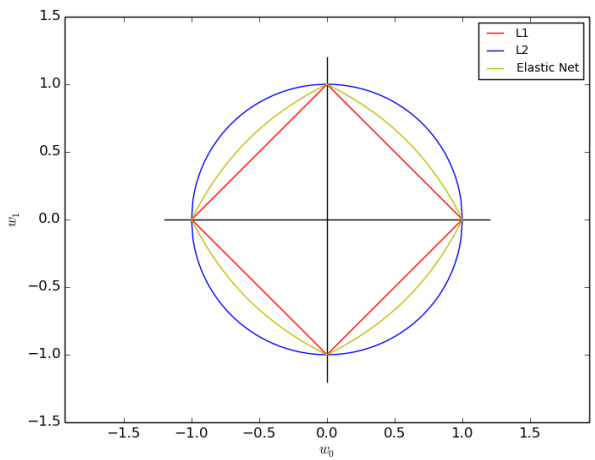


Figure 3: http://scikit-learn.sourceforge.net/

Descente de gradient proximale

Problème de minimisation

Nous avons vu des problèmes de minimisation de la forme

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} f(w) + g(w)$$

où f est une fonction de goodness-of-fit

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle w, x_i \rangle)$$

où ℓ est une fonction de perte et

$$g(w) = \frac{1}{C} \operatorname{pen}(w)$$

où $\operatorname{pen}(\cdot)$ est une pénalisation, par exemple $\operatorname{pen}(w) = \frac{1}{2}\|w\|_2^2$ (ridge) et $\operatorname{pen}(w) = \|w\|_1$ (Lasso)

Remarque : on oublie dans cette partie le paramètre b.

Gradient et hessienne

On veut minimiser

$$F(w) = f(w) + g(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \langle x_i, w\rangle) + \frac{1}{C}\,\mathrm{pen}(w)$$

Calculons le gradient et la hessienne de f

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell'(y_i, \langle x_i, w\rangle) x_i$$

$$\nabla^2 f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell''(y_i, \langle x_i, w\rangle) x_i x_i^\top$$

avec

$$\ell'(y, y') = \frac{\partial \ell'(y, y')}{\partial y'} \quad \text{et} \quad \ell''(y, y') = \frac{\partial^2 \ell'(y, y')}{\partial y'^2}$$

Convexité et L-régularité

Remarquons que f est convexe si et seulement si

$$y' \mapsto \ell(y_i, y')$$

l'est pour tout $i = 1, \ldots, n$.

Definition. On dit f est L-**régulière** si elle est continuement différentiable et si

$$\|\nabla f(w) - \nabla f(w')\|_2 \leq L\|w - w'\|_2 \quad \text{pour tout } w, w' \in \mathbb{R}^d$$

Si f est deux fois différentiable, c'est équivalent à supposer

$$\lambda_{\max}(\nabla^2 f(w)) \leq L \quad \text{for any } w \in \mathbb{R}^d$$

(la plus grande valeur propre de la hessienne de f est plus petite que L)

Cas particuliers : perte des moindres carrés

Pour la perte des moindres carrés (least-squares loss)

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n}(\langle x_i, w\rangle - y_i)x_i, \quad \nabla^2 f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top$$

donc

$$L = \frac{1}{n}\lambda_{\max}\left(\sum_{i=1}^{n} x_i x_i^\top\right)$$

Cas particuliers : perte logistique

Pour la perte logistique

$$\nabla f(w) = \frac{1}{n} \sum_{i=1}^{n} y_i (\sigma(y_i \langle x_i, w \rangle) - 1) x_i$$

et

$$\nabla^2 f(w) = \frac{1}{n} \sum_{i=1}^{n} \sigma(y_i \langle x_i, w \rangle)(1 - \sigma(y_i \langle x_i, w \rangle)) x_i x_i^\top$$

donc

$$L = \frac{1}{4n} \lambda_{\max} \Big(\sum_{i=1}^{n} x_i x_i^\top \Big)$$

Lemme de descente

Si f est L-régulière, alors

$$f(w) \leq f(w') + \langle \nabla f(w'), w - w' \rangle + \frac{L}{2} \|w - w'\|_2^2$$

pour tout $w, w' \in \mathbb{R}^d$

Preuve dans le cours d'optimisation

On a donc, autour du point w^k à l'itération k

$$f(w) \leq f(w^k) + \langle \nabla f(w^k), w - w^k \rangle + \frac{L}{2} \|w - w^k\|_2^2$$

pour tout $w \in \mathbb{R}^d$

Descente de gradient proximale

En considérant le problème de départ, on a donc à l'itération k

$$f(w) + g(w) \leq f(w^k) + \langle \nabla f(w^k), w - w^k \rangle + \frac{L}{2}\|w - w^k\|_2^2 + g(w)$$

et

$$\begin{aligned}
&\operatorname*{argmin}_{w \in \mathbb{R}^d} \left\{ f(w^k) + \langle \nabla f(w^k), w - w^k \rangle + \frac{L}{2}\|w - w^k\|_2^2 + g(w) \right\} \\
&\quad = \operatorname*{argmin}_{w \in \mathbb{R}^d} \left\{ \frac{L}{2} \left\| w - \left(w^k - \frac{1}{L}\nabla f(w^k) \right) \right\|_2^2 + g(w) \right\} \\
&\quad = \operatorname*{argmin}_{w \in \mathbb{R}^d} \left\{ \frac{1}{2} \left\| w - \left(w^k - \frac{1}{L}\nabla f(w^k) \right) \right\|_2^2 + \frac{1}{L} g(w) \right\} \\
&\quad = \ ????
\end{aligned}$$

Opérateur proximal

Pour tout $g : \mathbb{R}^d \to \mathbb{R}$ convexe (pas forcément différentiable), et tout $w \in \mathbb{R}^d$, on définit

$$\text{prox}_g(w) = \underset{w' \in \mathbb{R}^d}{\text{argmin}} \left\{ \frac{1}{2} \| w - w' \|_2^2 + g(w') \right\}$$

Prox de la pénalité ridge

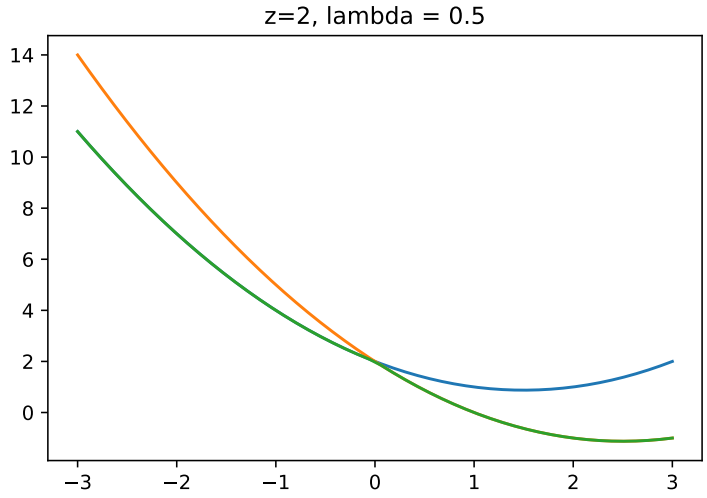
Calculer l'opérateur proximal de la pénalité ridge.

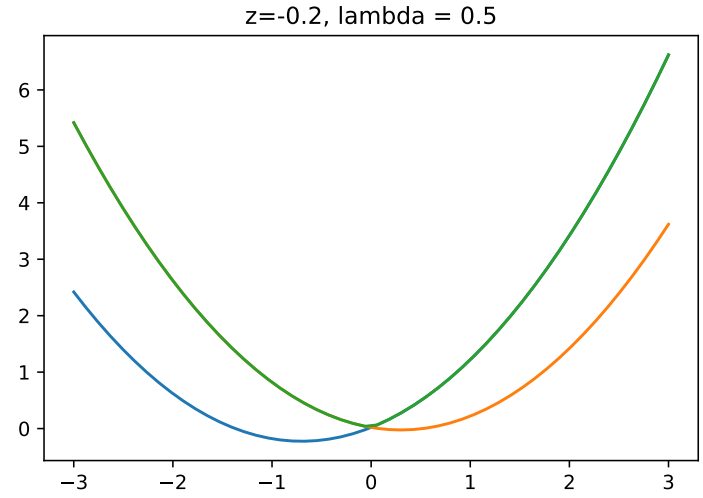
Calcul direct du prox du LASSO (1)

Considérons le problème de minimisation

$$\min_{z' \in \mathbb{R}} \frac{1}{2}(z' - z)^2 + \lambda |z'|$$

pour $\lambda > 0$ et $z \in \mathbb{R}$.





Calcul direct du prox du LASSO (2)

- La dérivée sur $\mathbb{R}+ +^\star$: $z' - z + \lambda$, en $0+$ $d_+ = -z + \lambda$
- La dérivée en $\mathbb{R}_+^\star$: $z' - z - \lambda$, en $0-$ $d_- = -z - \lambda$

Soit z_* la solution, elle vérifie

- $z_* = 0$ ssi $d_+ \geq 0$ et $d_- \leq 0$, soit $|z| \leq \lambda$
- $z_* \geq 0$ ssi $d_+ \leq 0$, soit $z \geq \lambda$ et $z_* = z - \lambda$
- $z_* \leq 0$ ssi $d_- \geq 0$, soit $z \leq -\lambda$ et $z_* = z + \lambda$

donc

$$z_* = \text{sign}(z)(|z| - \lambda)_+$$

On l'appelle **l'opérateur de seuillage doux** (soft-thresholding operator).

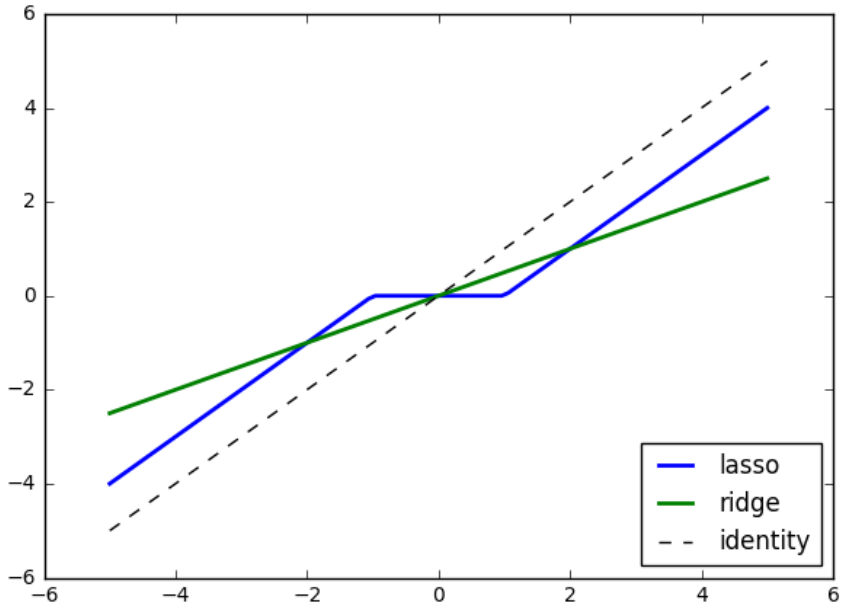
Calcul direct du prox du LASSO (3)

$$\operatorname*{argmin}_{z' \in \mathbb{R}} \frac{1}{2}(z' - z)^2 + \frac{1}{C}|z'| = \operatorname{sign}(z)\left(|z| - \frac{1}{C}\right)_+$$

donc

$$\operatorname*{argmin}_{w' \in \mathbb{R}^d} \frac{1}{2}\|w' - w\|_2^2 + \frac{1}{C}\|w'\|_1 = \operatorname{sign}(w) \odot \left(|w| - \frac{1}{C}\right)_+ .$$

Exemple avec $C = 1$



Descente de gradient proximale (PGD)

- **Input**: initialisation w^0, constante de Lipschitz $L > 0$ pour ∇f,
- pour $k = 1, 2, \ldots$ jusqu'à *convergence* faire

$$w^k \leftarrow \text{prox}_{g/L}\left(w^{k-1} - \frac{1}{L}\nabla f(w^{k-1})\right)$$

- **Renvoyer** w^k

Pour le Lasso

$$\hat{w} \in \underset{w \in \mathbb{R}^d}{\text{argmin}} \left\{ \frac{1}{2n}\|y - Xw\|_2^2 + \lambda \|w\|_1 \right\},$$

l'itération est donnée par

$$w^k \leftarrow S_{\lambda/L}\left(w^{k-1} - \frac{1}{Ln}X^\top (Xw^{k-1} - y)\right),$$

où S_λ est l'opérateur de seuillage doux.

Exercices

Avec l'intercept b

Récrire l'algorithme de descente de gradient proximale quand ℓ dépend à la fois de w et de b, c'est-à-dire pour le problème de minimisation

$$\hat{w}, \hat{b} \in \operatorname*{argmin}_{w \in \mathbb{R}^d, b \in \mathbb{R}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b) + \frac{1}{C} \operatorname{pen}(w) \right\}.$$

Elastic-net

Récrire l'algorithme de descente de gradient proximale pour la pénalité elastic-net

$$\operatorname{pen}(w) = \frac{1-\alpha}{2} \|w\|_2^2 + \alpha \|w\|_1$$

Point d'étape

On sait calculer

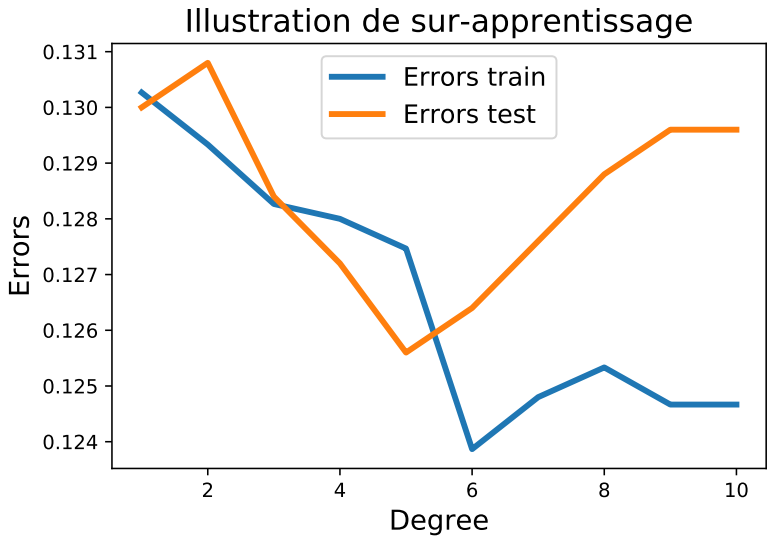
$$\hat{w}, \hat{b} \in \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b) + \frac{1}{C} \operatorname{pen}(w) \right\}.$$

dans les cas du LASSO $\operatorname{pen}(w) = \|w\|_1$ et de la ridge $\operatorname{pen}(w) = \frac{1}{2}\|w\|_2^2$ pour une valeur de C ou de $\lambda = 1/C$. Il reste donc à choisir $C > 0$ ou $\lambda > 0$.

Cross-validation

Sur-apprentissage / sur-ajustement / over-fitting

Sur le jeu de données d'exemple `linear` j'ai ajouté des features en prenant des polynômes des features initiales.



But de l'apprentissage statistique

- Le but de l'apprentissage supervisé (dans le cas de la classification) est en fait de trouver le classifieur qui minimise l'erreur de généralisation

$$c^*_{\text{generalisation}} \in \operatorname*{argmin}_c \mathbb{E}\Big(\ell(Y_+, c(X_+))\Big)$$

ou, dans les cas étudiés dans ce chapitre:

$$w^*_{\text{generalisation}} \in \operatorname*{argmin}_{w\in\mathbb{R}^d} \mathbb{E}(\ell(Y_+, \langle X_+, w\rangle))$$

- Pourtant nous définissions

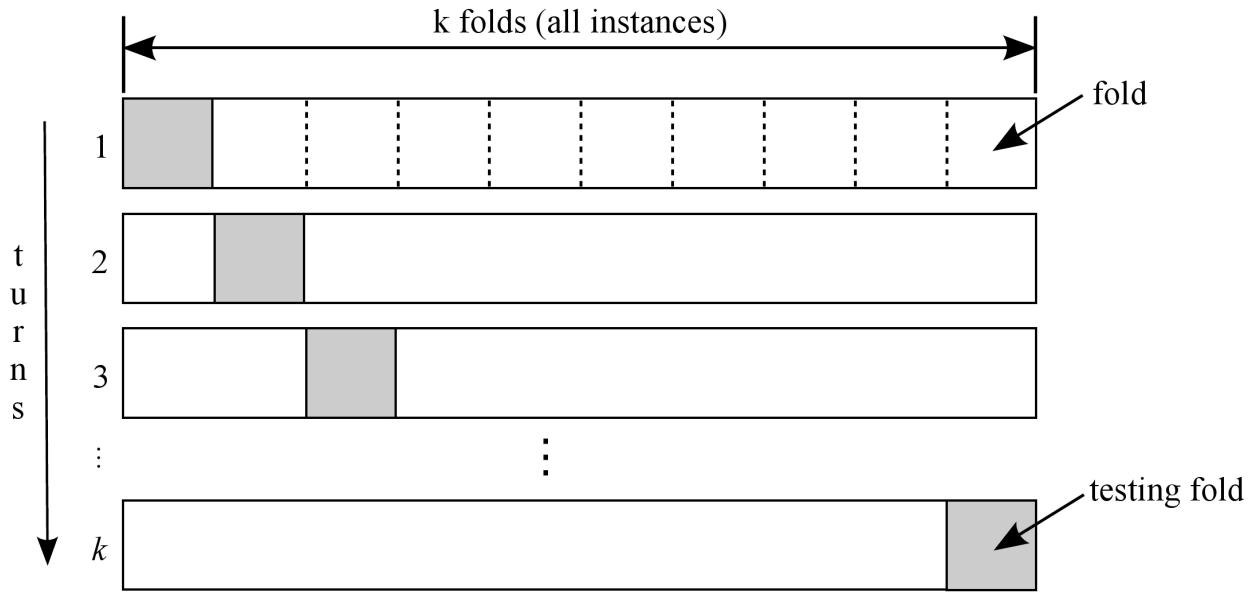
$$\hat{w}, \hat{b} \in \operatorname*{argmin}_{w\in\mathbb{R}^d, b\in\mathbb{R}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \langle x_i, w\rangle + b) + \frac{1}{C}\,\text{pen}(w) \right\}.$$

- On doit donc trouver une valeur de C qui rend petite l'erreur de généralisation.
- On va utiliser la cross-validation en montrant comment elle permet d'estimer l'erreur de généralisation.

Take home message il n'y a pas de machine learning sans cross-validation !

Cross-validation V-Fold

- On prend $V = 5$ ou $V = 10$. On choisit une partition aléatoire $I_1, \ldots, I_V$ de $\{1, \ldots, n\}$, où $|I_v| \approx \frac{n}{V}$ pour tout $v = 1, \ldots, V$



On choisit une grille

$$\mathcal{C} = \{C_1, \dots C_K\}$$

de valeurs possibles pour C. Pour tout $v = 1, \dots, V$

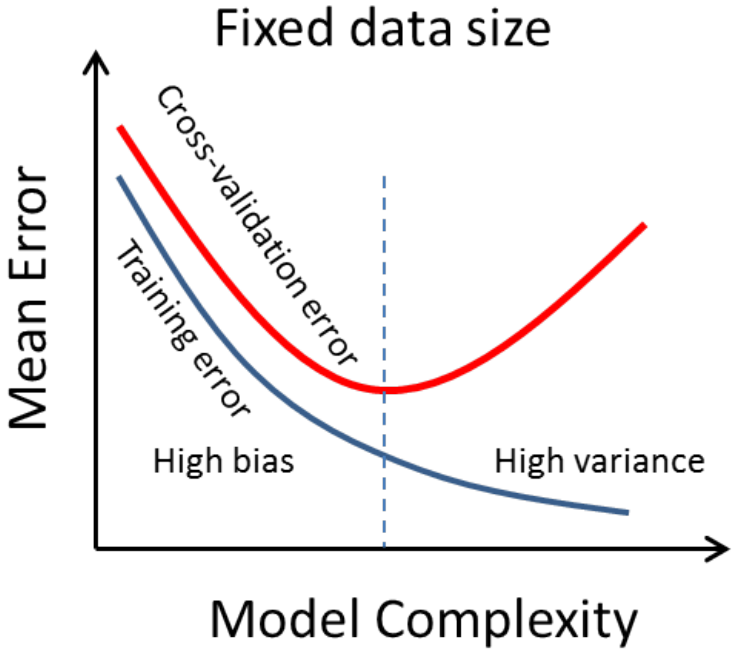
- Posons $I_{v,\texttt{train}} = \cup_{v' \neq v} I_{v'}$ et $I_{v,\texttt{test}} = I_v$
- Pour tout $C \in \mathcal{C}$, on cherche

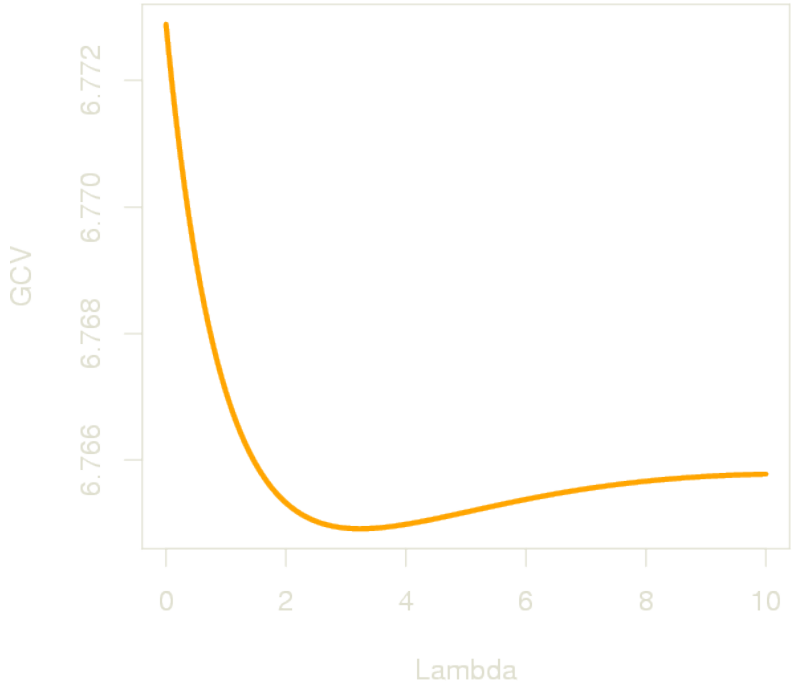
$$\hat{w}_{v,C} \in \operatorname*{argmin}_{w} \left\{ \frac{1}{|I_{v,\texttt{train}}|} \sum_{i \in I_{v,\texttt{train}}} \ell(y_i, \langle x_i, w \rangle) + \frac{1}{C} \operatorname{pen}(w) \right\}$$

On pose

$$\hat{C} \in \operatorname*{argmin}_{C \in \mathcal{C}} \sum_{v=1}^{V} \sum_{i \in I_{v,\texttt{test}}} \ell(y_i, \langle x_i, \hat{w}_{v,C} \rangle)$$

Remarque on peut utiliser d'autres pertes ou métriques pour choisir $\hat{C}$





- Erreur visible/erreur empirique/training error:

$$C \mapsto \sum_{v=1}^{V} \sum_{i \in I_{v,\texttt{train}}} \ell(y_i, \langle x_i, \hat{w}_{v,C} \rangle)$$

- Erreur de test/de validation/de cross-validation/testing error

$$C \mapsto \sum_{v=1}^{V} \sum_{i \in I_{v,\texttt{test}}} \ell(y_i, \langle x_i, \hat{w}_{v,C} \rangle)$$

Métriques en classification

Métriques standard classification (1)

- Precision, Recall, F-Score, AUC

Pour chaque individu i nous avons

- son vrai label y_i
- son label prédit $\hat{y}_i$

On peut construire **la matrice de confusion**

		Predicted Class: Yes	Predicted Class: No
Actual Class	Yes	TP	FN
	No	FP	TN

with

$$\begin{aligned} \text{TP} &= \sum_{i=1}^{n} \mathbb{1}_{y_i=1,\hat{y}_i=1} \\ \text{TN} &= \sum_{i=1}^{n} \mathbb{1}_{y_i=-1,\hat{y}_i=-1} \\ \text{FN} &= \sum_{i=1}^{n} \mathbb{1}_{y_i=1,\hat{y}_i=-1} \\ \text{FP} &= \sum_{i=1}^{n} \mathbb{1}_{y_i=-1,\hat{y}_i=1} \end{aligned}$$

avec yes $= 1$ et no $= -1$

Métriques standard classification (2)

$$\text{Precision} = \frac{\text{TP}}{\#(\text{predicted P})} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\#(\text{real P})} = \frac{\text{TP}}{\text{TP} + \text{FN}} =$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{F-Score} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Un peu de vocabulaire

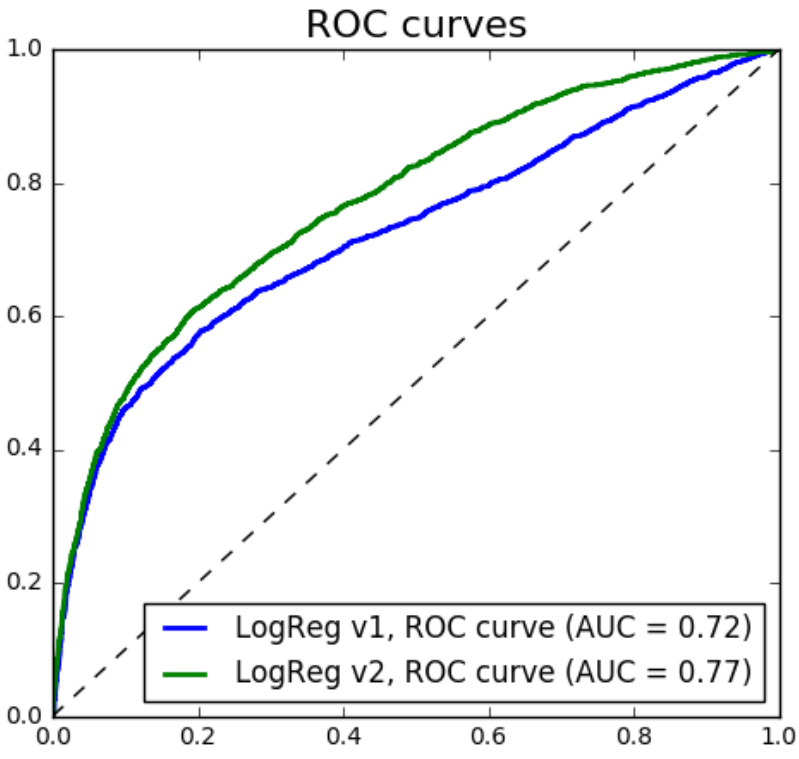
- Recall = Sensitivity
- False-Discovery Proportion FDP $= 1 -$ Precision

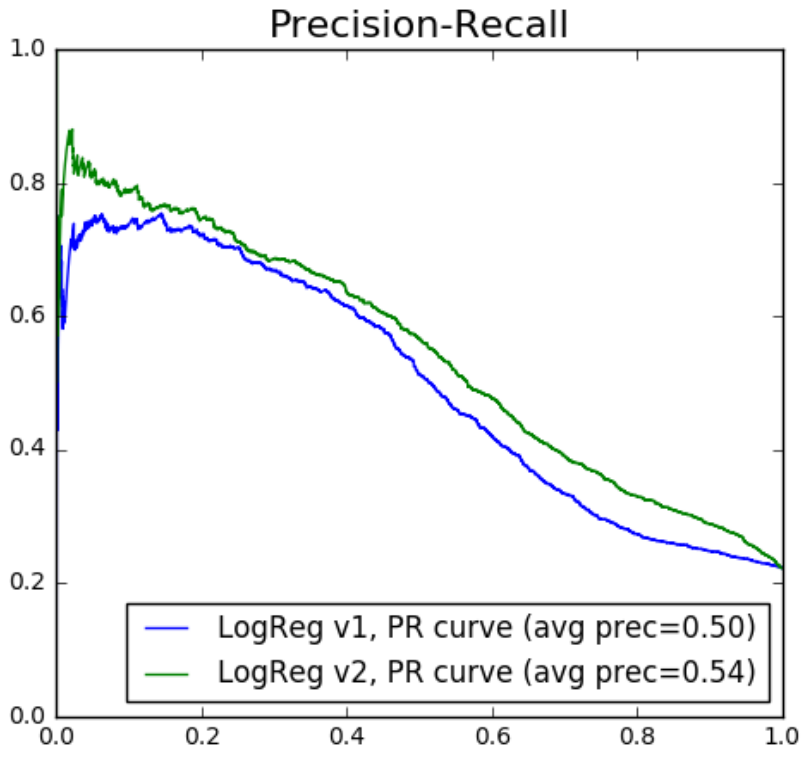
Courbe ROC

- On part des probalitités estimées $\hat{\pi}_1(x_i) = \hat{\mathbb{P}}(Y = 1 | X = x_i)$
- Chaque point A_t de la courbe a pour coordonnées $(\text{FPP}_s, \text{Recall}_s)$, où FPP_s et Recall_s sont les FPP et le recall de la matrice de confusion obtenue avec la règle de classification

$$\hat{Y}_i = \begin{cases} 1 & \text{si } \hat{\pi}_1(x_i) \geq s \\ -1 & \text{sinon} \end{cases}$$

  pour un seuil s variant dans $[0, 1]$
- l'AUC est alors l'aire sous la courbe ROC.


ROC curves
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
LogReg v1, ROC curve (AUC = 0.72)
LogReg v2, ROC curve (AUC = 0.77)


Precision-Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
LogReg v1, PR curve (avg prec=0.50)
LogReg v2, PR curve (avg prec=0.54)