Machine Learning

Ouattara Mory

IDSI- INP-HB

2022-2023

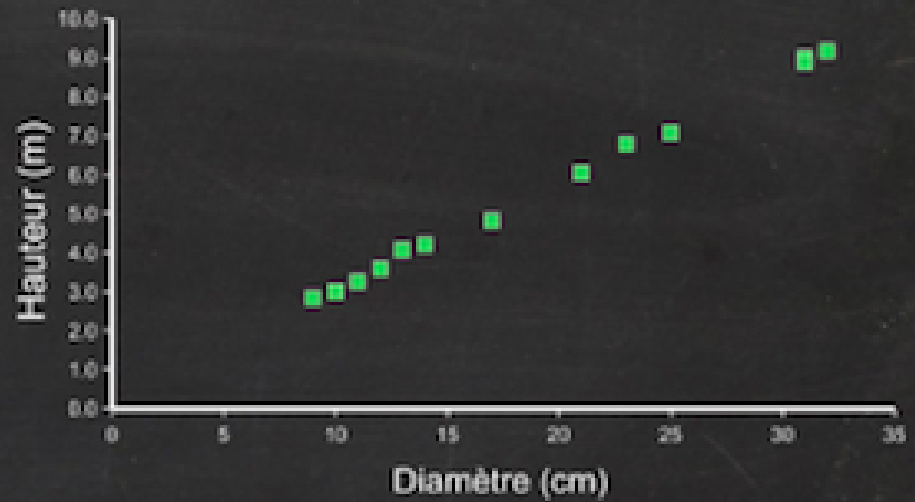
Par l'exemple : En prédiction

Diamètre (cm)	Hauteur (m)
10	2.8
17	4.6
14	4.0
9	2.6
21	5.8
25	6.9
31	8.8
12	3.4
23	6.6
32	9.0
31	8.7
11	3.0
13	3.8

Une façon d'intéragir avec l'environnement

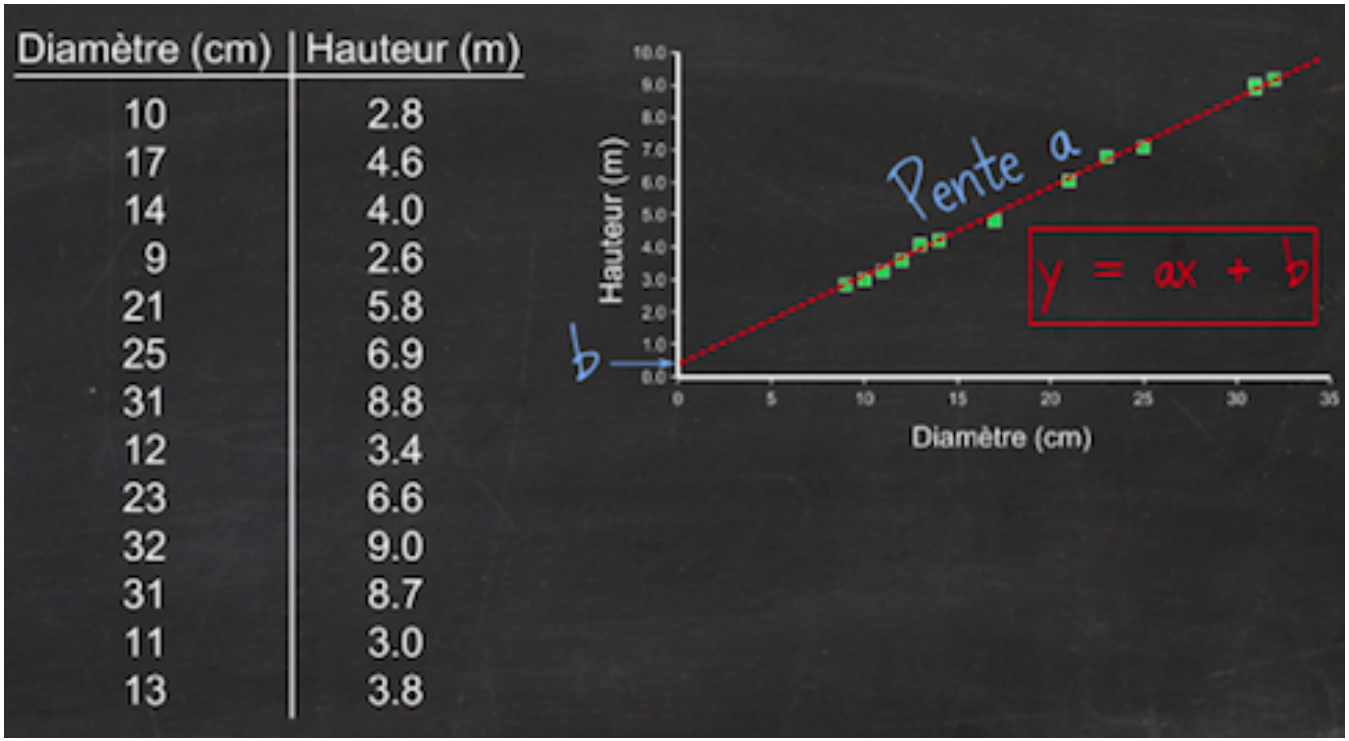
Par l'exemple

Diamètre (cm)	Hauteur (m)
10	2.8
17	4.6
14	4.0
9	2.6
21	5.8
25	6.9
31	8.8
12	3.4
23	6.6
32	9.0
31	8.7
11	3.0
13	3.8



Projection des données

En prédiction

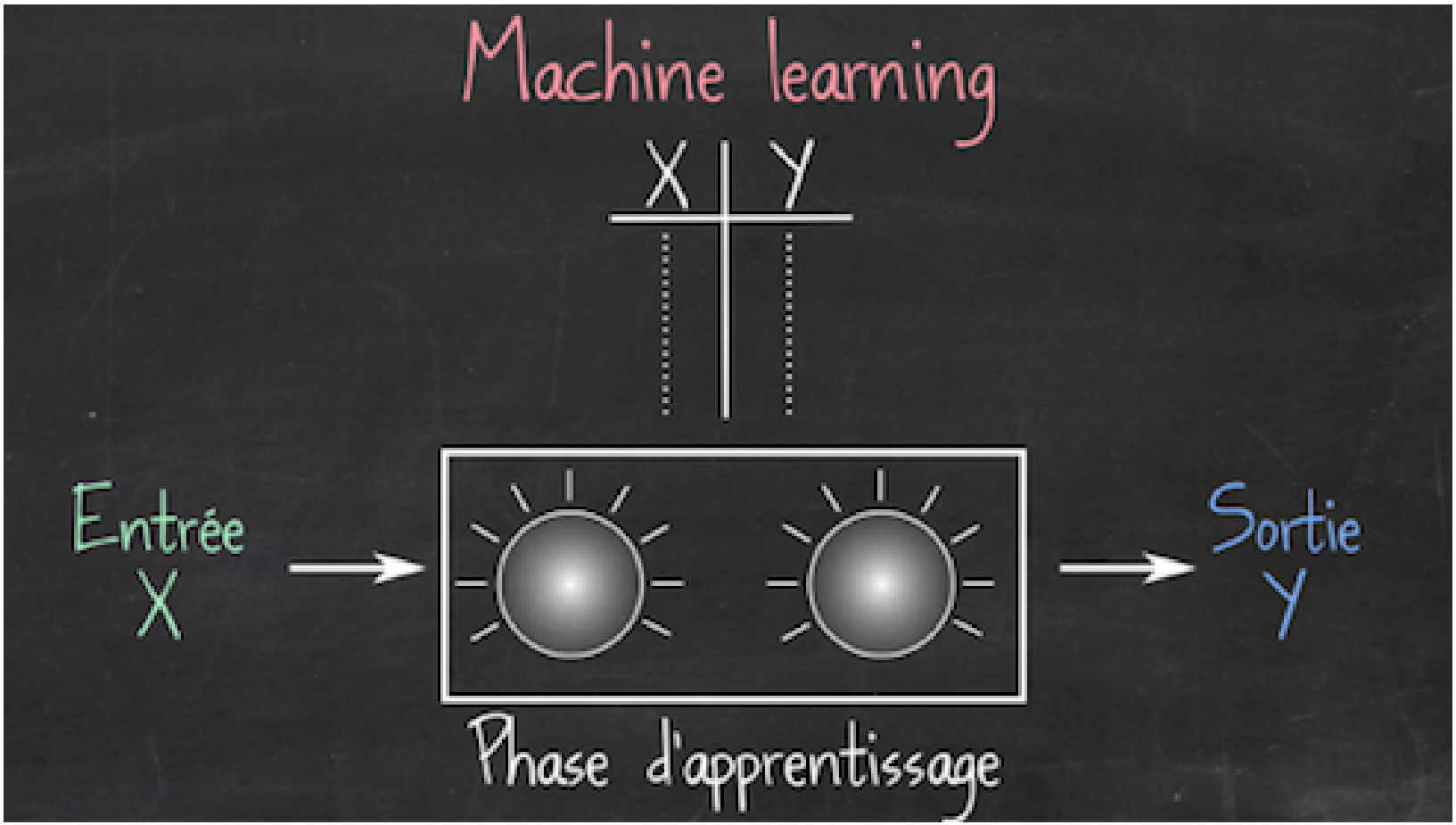


Un modèle, le plus souvent paramétrique $Y = ax + b$ et fonction de coût :

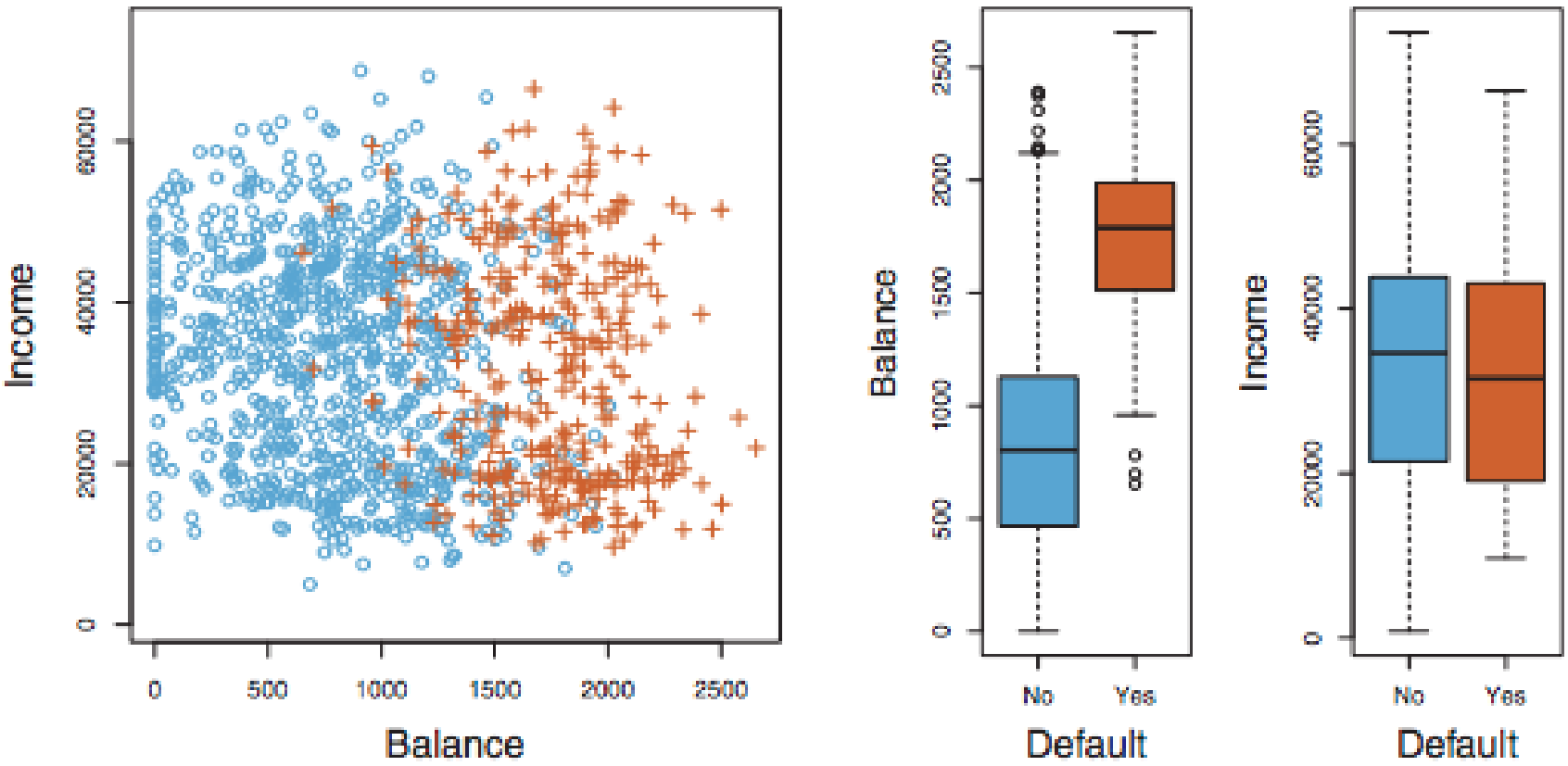
$$\sum_i (y_i - ax_i - b)^2$$

Et un algorithme de résolution directe ou itérative.

En prédiction : Résumé



Exemple

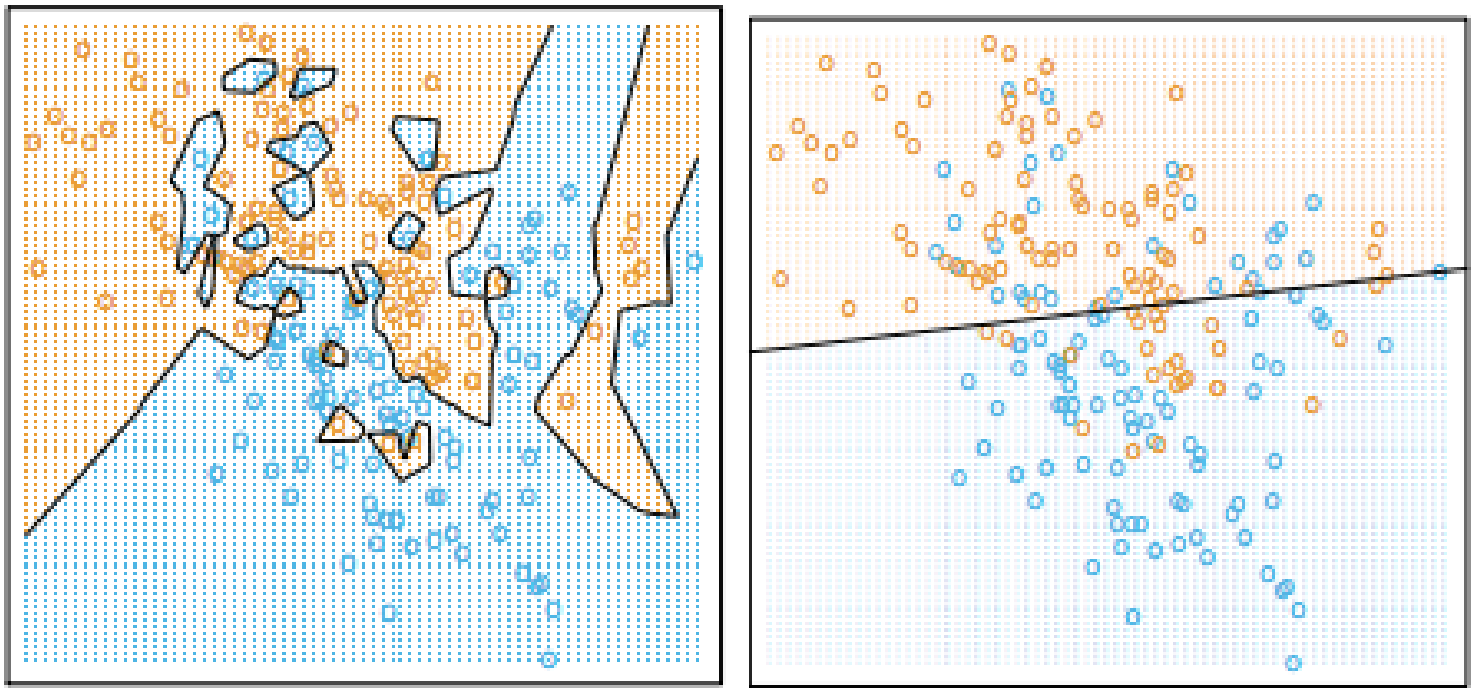


Problème de classification : Considérons un jeu de données où chaque observation peut appartenir à une classe parmi **K** classes.

On souhaite ensuite classer un individu dont on ignore la classe.

Exemple

Comment classifier une nouvelle observation ?



Plusieurs possibilités :

- Tenir compte du voisinage du nouveau point avec les points déjà labellisés.
- Construire une frontière de décision.

Exemple

Disposons de représentation manuscrite (graphique) des chiffres 1 et 7

Comment apprendre à reconnaitre les chiffres 1 et 7.

- Apprendre un classifieur
- Tester le classifieur sur de nouvelles images des chiffres 1 et 7

Objectif général

Un objectif de **modélisation** décliné en sous-objectifs à définir clairement préalablement à une étude :

- **Explorer, Représenter, Décrire,** les variables, leurs liaisons et positionner les observations de l'échantillon
- **Expliquer ou Tester** l'influence d'une variable ou facteur dans un modèle supposé connu a priori
- **Prévoir et Sélectionner** un meilleur ensemble de prédicteurs

Nous nous concentrons sur les stratégies de choix de modèle parmi un ensemble plus ou moins complexe, de choix de méthode ... en se focalisant sur les pratiques ou méthodes à l'interface de l'apprentissage machine et de la Statistique

Formalisme

En présence d'une variable à expliquer Y qui a été, conjointement avec X, observée sur les mêmes objets. On definit l'ensemble suivant des observations :

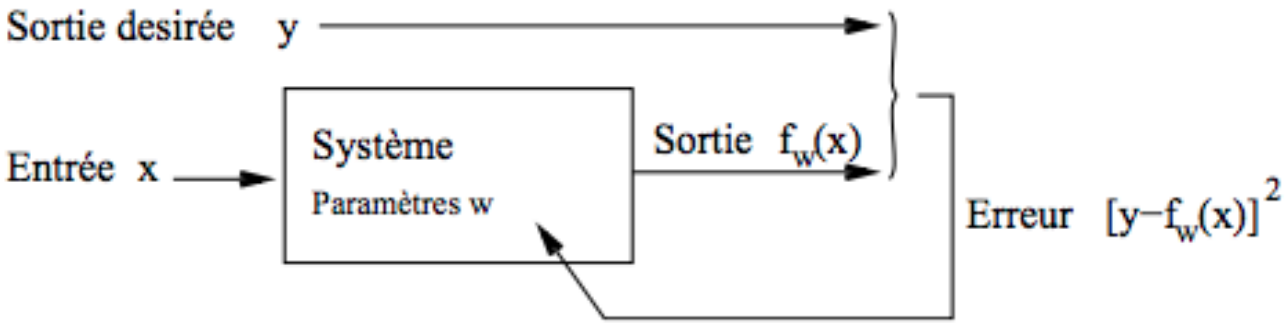
$$d_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

on cherche à **expliquer/prédire** les sorties $y_i \in \mathcal{Y}$ à partir des entrées $x_i \in \mathcal{X}$

On cherche donc une fonction f **susceptible, au mieux** selon **un critère à définir**, de **reproduire** Y.

$$Y = f(X) + \varepsilon$$

où ε symbolise le bruit ou erreur de mesure



Formalisme

En présence d'une variable à expliquer Y qui a été, conjointement avec X, observée sur les mêmes objets. On definit l'ensemble suivant des observations :

$$d_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

on cherche à **expliquer/prédire** les sorties $y_i \in \mathcal{Y}$ à partir des entrées $x_i \in \mathcal{X}$

On cherche donc une fonction f **susceptible, au mieux** selon **un critère à définir**, de **reproduire** Y.

$$Y = f(X) + \varepsilon$$

où ε symbolise le bruit ou erreur de mesure

Le plus souvent, on utilise une fonction de perte $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ telle que

$$\begin{cases} \ell(y, y') = 0 & \text{si } y = y' \\ \ell(y, y') > 0 & \text{si } y \neq y' \end{cases}$$

Fonctions de lien

Sans être exhaustif

(l_1) Lien linéaire : $f(x) = \alpha + \beta x$ avec β et $x \in \mathbb{R}^p$ et α est un élément de $\mathbb{R}$

(l_2) Lien polynomiale $f(x) = \alpha + \beta_1 x + \beta_2 x^2 + ...$ avec β et $x \in \mathbb{R}^p$

(l_3) Lien Logistique : $f(x) = g(\beta x + \alpha)$ avec $g(u) = \frac{1}{1+e^{-u}}$

Fonctions de Perte

Sans être exhaustive

Soit $z = f(x)$ la prédiction de x par la fonction de lien f et y la valeur observée.

(p_1) $\ell(y, z) = (y - z)^2$ le carré de l'erreur

(p_2) $\ell(y, z) = |y - z|$ l'erreur absolue

(p_3) $\ell(y, z) = -yln(z) - (1 - y)ln(1 - z)$ erreur en régression logistique $y \in [0, 1]$ et $z \in \{0, 1\}$

(p_4) Classification binaire ($\mathcal{Y} = \{-1, 1\}$) :

$$\begin{aligned} \ell : \{-1, 1\} \times \{-1, 1\} &\to \mathbb{R}^+ \\ (y, y') &\mapsto 1_{y \neq y'} \end{aligned}$$

Le risque pour une fonction de prévision ou règle de prévision $g : \mathcal{X} \to \{-1, 1\}$ est alors donné par

$$\mathcal{R}(g) = \mathrm{E}\left(1_{g(X) \neq Y}\right) = \mathrm{P}(g(X) \neq Y)$$

Fonction de score

On reste dans un cadre de classification binaire ($\mathcal{Y} = \{-1, 1\}$).

On cherche une fonction $S : \mathcal{X} \to \mathbb{R}$ telle que $S(x) = \mathrm{P}(Y = 1 \mid X = x)$

Une telle fonction est appelée fonction de score : plutôt que de prédire directement le groupe d'un nouvel individu $x \in \mathcal{X}$, on lui donne une note $S(x)$

- S(x) est élevée si il a des "chances" d'être dans le groupe 1
- S(x) est faible si il a des "chances" d'être dans le groupe -1 ;

Approche statistique

On suppose que les données $d_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ sont des réalisations d'un n-échantillon $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ de loi inconnue.

Les X_i sont des variables aléatoires à valeurs dans $\mathcal{X}$, les Y_i dans $\mathcal{Y}$.

Le plus souvent on supposera que les couples $(X_i, Y_i), i = 1, \ldots, n$ sont i.i.d de loi P.

Performance d'une fonction de prévision

Etant donné une fonction de perte $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, la performance d'une fonction (mesurable) de prévision $f : \mathcal{X} \to \mathcal{Y}$ est mesurée par

$$\mathcal{R}(f) = \mathrm{E}[\ell(Y, f(X))]$$

où (X, Y) est indépendant des (X_i, Y_i) et de même loi P.

$R(f)$ **est appelé risque ou erreur de généralisation de f**

Fonction de prévision optimale

Aspect théorique

Pour une fonction de perte $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ donnée, le problème théorique consiste à trouver

$$f^\star \in \underset{f}{\operatorname{argmin}} \mathcal{R}(f)$$

Une telle fonction $f^\star$ (si elle existe) est appelée fonction de prévision optimale pour la perte ℓ

Fonction de prévision optimale

Aspect pratique

La fonction de prévision optimale $f^\star$ dépend le plus souvent de la loi P des (X, Y) qui est en pratique inconnue.

Objectif trouver un estimateur $f_n = f_n(., \mathcal{D}_n)$ tel que $\mathcal{R}(f_n) \approx \mathcal{R}(f^\star)$.

Un algorithme de prévision est représenté par une suite $(f_n)_n$ d'applications (mesurables) telles que pour $n \geq 1$, $f_n : (\mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^n) \to \mathcal{Y}$

On dit que la suite $(f_n)_n$ est universellement consistante si $\forall P$

$$\lim_{n \to \infty} \mathcal{R}(f_n) = \mathcal{R}(f^\star)$$

Choix de la fonction de perte

Le cadre mathématique développé précédemment sous-entend qu'une fonction est performante (voire optimale) vis-à-vis d'un critère (représenté par la fonction de perte ℓ)).

Un algorithme de prévision performant pour un critère ne sera pas forcément performant pour un autre.

Conséquence pratique

Avant de s'attacher à construire un algorithme de prévision, il est capital de savoir mesurer la performance d'un algorithme de prévision.

Estimation du risque

n observations $(X_1, Y_1), \ldots, (X_n, Y_n)$ i.i.d à valeurs dans $\mathcal{X} \times \mathcal{Y}$.

Objectif

Etant donnée une fonction de perte $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$

on cherche un algorithme de prévision $f_n(x) = f_n(x, \mathcal{D}_n)$ qui soit "proche" de l'oracle $f^\star$ défini par

$$f^\star \in \underset{f}{\operatorname{argmin}} \mathcal{R}(f)$$

où $\mathcal{R}(f) = \mathsf{E}[\ell(Y, f(X))]$.

Etant donné un algorithme f_n, que vaut son risque $\mathcal{R}(f_n)$?

Risque empirique

- La loi de (X, Y) étant inconnue en pratique, il est impossible de calculer $\mathcal{R}(f_n) = \mathrm{E}[\ell(Y, f_n(X))]$.

- Première approche : $\mathcal{R}(f_n)$ étant une espérance, on peut l'estimer (LGN) par sa version empirique

$$\mathcal{R}_n(f_n) = \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, f_n(X_i))$$

problème

- L'échantillon $\mathcal{D}_n$ a déjà été utilisé pour construire l'algorithme de prévision $f_n \Longrightarrow$ La LGN ne peut donc s'appliquer !
- Conséquence : $\mathcal{R}_n(f_n)$ conduit souvent à une sous-estimation de $\mathcal{R}(f_n)$.

Solution ? ? ?

Apprentissage - Validation

Elle consiste à séparer l'échantillon D_n en :

1. un échantillon d'apprentissage $\mathcal{D}_{n,app}$ pour construire f_n ;
2. un échantillon de validation $\mathcal{D}_{n,test}$ utilisé pour estimer le risque de f_n

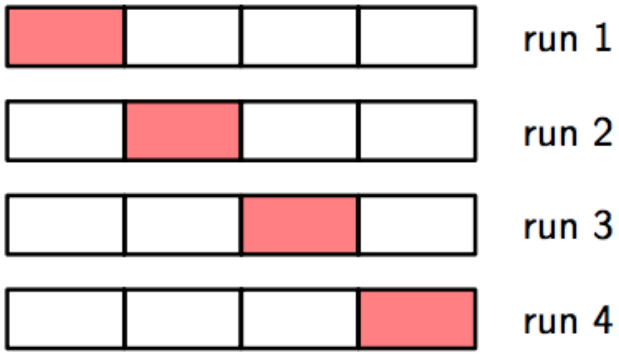
Algorithme

Entrées. $\mathcal{D}_n$: données, $\{\mathcal{A}, \mathcal{V}\}$: partition de $\{1, \ldots, n\}$

- Construire l'algorithme de prédiction sur $\mathcal{D}_{n,app} = \{(X_i, Y_i) : i \in \mathcal{A}\}$, on le note $f_{n,app}$;
- Calculer $\widehat{\mathcal{R}}_n(f_n) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \ell(Y_i, f_{n,\text{ app }}(X_i))$.

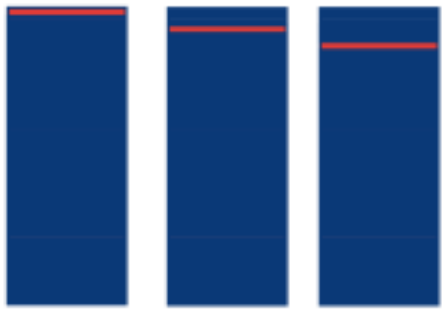
Séparer ensemble d'entraînement et de test

- Deux ensembles séparées
- **N-Fold Cross Validation** (validation à N-plis croisés)
  - D est divisé en N sous-ensemble D_n
  - test : on choisit un sous-ensemble
  - entraînement : on utilise les N-1 autres
  - on réitère en choisissant un nouveau sous-ensemble de test parmi les N possibles
  - on calcul la moyenne des indices



Séparer ensemble d'entraînement et de test

- Deux ensembles séparées
- **Leave-one-out Cross-Validation :**
  - cas limite quand N=le nombre de données
  - test : on choisit une donnée
  - entraînement : on utilise les N-1 autres données
  - on réitère en choisissant une nouvelle donnée de test parmi les N possibles
  - on calcul la moyenne des indices



Validation croisée K-blocs

- Plus adapté que la technique apprentissage/validation lorsqu'on a peu d'observations.
- Le choix de K doit être fait par l'utilisateur (souvent $K = 10$).

Leave one out

Lorsque $K = n$, on parle de validation croisée leave one out ; Le risque est alors estimé par

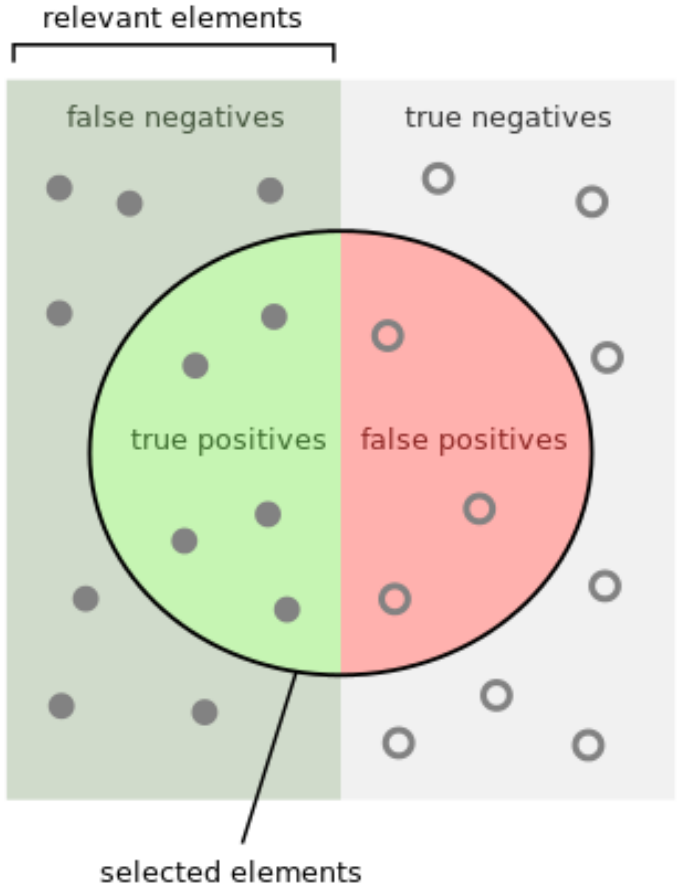
$$\widehat{\mathcal{R}}_n(f_n) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(Y_i, f_n^i(X_i)\right)$$

où f_n^i désigne l'algorithme de prévision construit sur $\mathcal{D}_n$ amputé de la i-ème observation.

Indices de performance

Calcul des TP, FP, TN, FN : pour deux classes ($c_1 = " + ", c_2 = " - "$)

- **True Positif (TP)** : cardinal de données + détectées correctement (True) comme +
- **False Negatif (FN)** : cardinal de données + détectées faussement (False) comme -
- **False Positif (FP)** : cardinal de données - détectées faussement (False) comme +
- **True Negatif (TN)** : cardinal de données - détectées correctement (True) comme -



Matrice de confusion

		Prédiction	
		Positif	Négatif
Vérité	Positif	TP	FN
	Négatif	FP	TN

Calcul des indices

Rappel ou sensibilité

Pour une classe c_k

$$Rappel_k = \frac{\sharp \text{ données correctement attribué à } c_k}{\sharp \text{ données vraiment } c_k} \text{ et } Rappel = \frac{TP}{TP + FN}$$

Précision

Pour une classe c_k

$$\text{Précision}_k = \frac{\sharp \text{ données correctement attribué à } c_k}{\sharp \textbf{de données attribuées} c_k} \text{ et Précision} = \frac{TP}{TP + FP}$$

Calcul des indices

Accuracy

Mesure les performances globales indépendamment de la distribution

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

F-Mesure

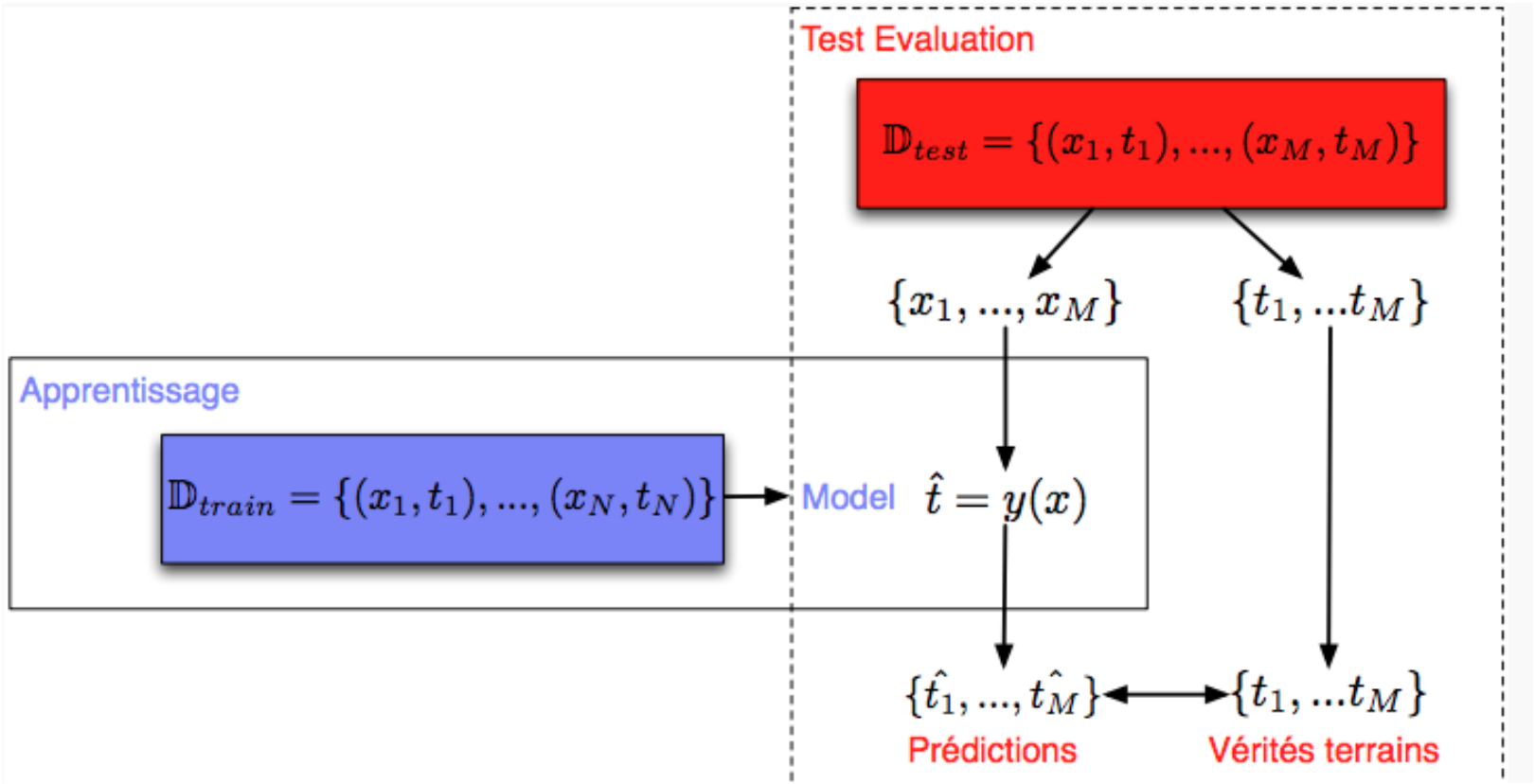
Prend en compte simultanément le Rappel et la Précision

$$F - Mesure = 2.\frac{Prcision * Rappel}{Prcision + Rappel}$$

Evaluation d'un système de classification

- L'ensemble de test $\mathbb{D}_{test}$ doit être différent de l'ensemble d'entraînement $\mathbb{D}_{train}$

  On test la généralisation du modèle $y(x)$
- On compare les prédictions de classes y_m aux "vérités terrains" de classes y_m
  Comment compare-t'on ?



Généralisation

Apprentissage

- Apprendre un modèle (génératif ou discriminant) à partir des observations X et des valeurs à prédire Y
- Le modèle doit permettre une bonne prédiction de Y en fonction des observations X

Généralisation

- Capacité du modèle à prédire correctement des valeurs Y en fonction de X en dehors de l'ensemble d'apprentissage
- Sur-apprentissage (over-fitting)
- En pratique on évalue les performances d'un modèle appris en séparant : Ensemble d'entrainement et Ensemble de test

Résumé

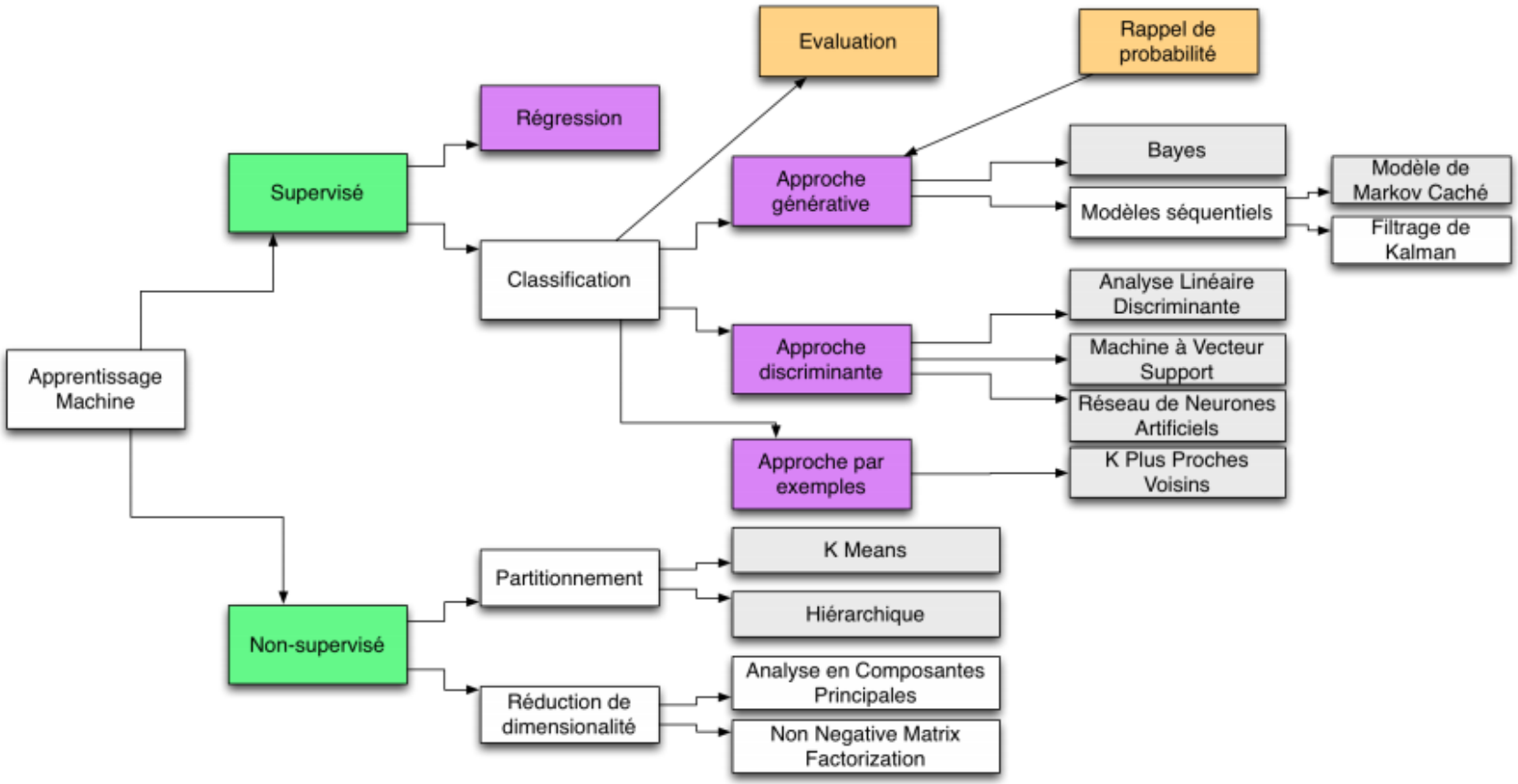


Figure – Geoffroy Peeters

Paradigme d'apprentissage

- Régression régularisée
- Arbres de décision
- Bagging, Boosting, Forêts aléatoire
- Rétropropagation du gradient sur réseau neuronal à couches
- Réseau de neurones : Cartes topologiques de Kohonen
- Support Vector Machines

Régression Régularisée

Rappel des notations

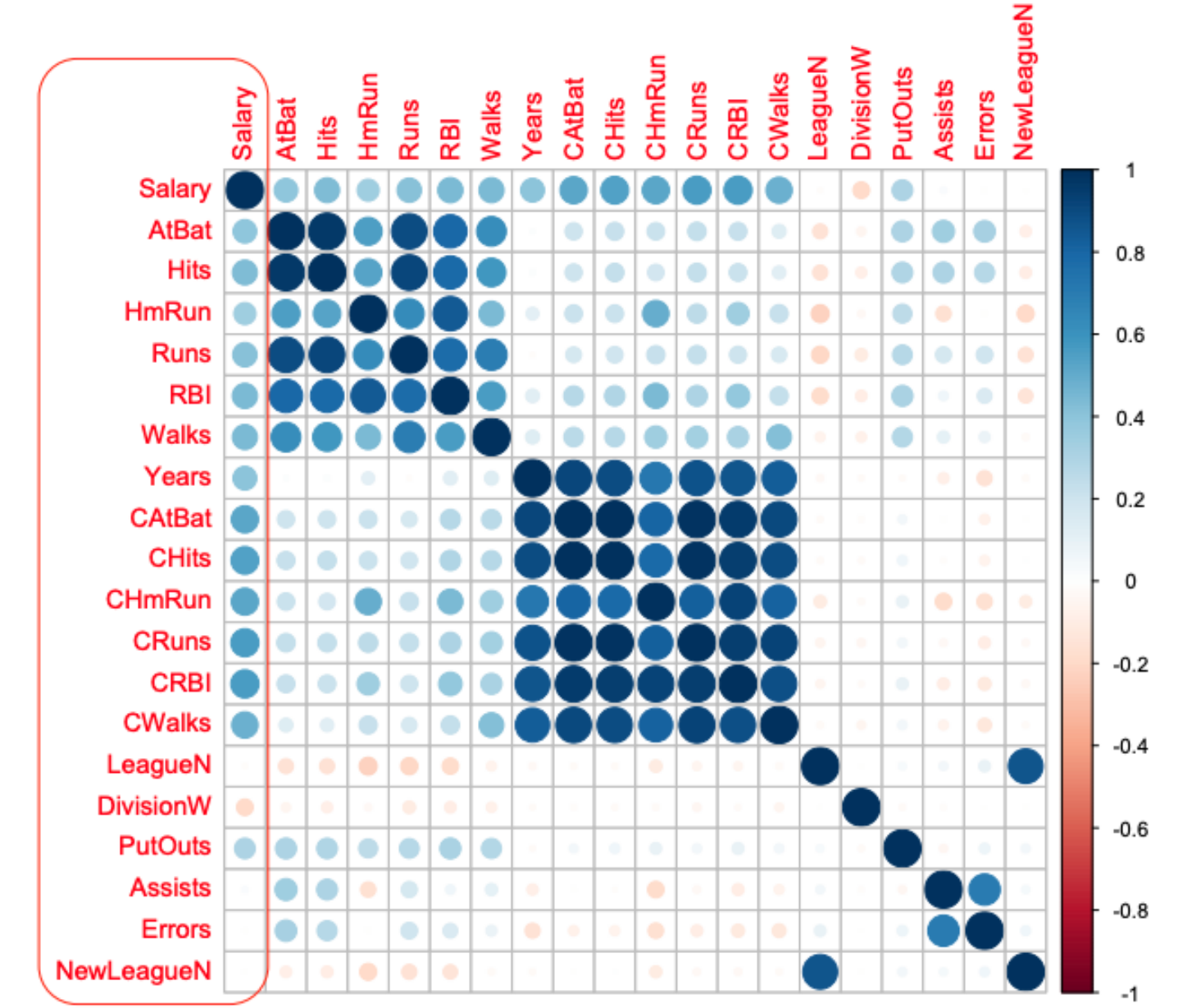
- **Données observées** de type entrée-sortie : $\mathcal{D}_n = (x_1, y_1), ..., (x_n, y_n)$ avec $x_i \in \mathbb{R}^p$, $y_i \in \mathbb{R}$ pour $i = 1, \ldots, n$.

- **Objectif** : prédire la sortie y associée à une nouvelle entrée x, sur la base de $\mathcal{D}_n$

Exemple

	Salary	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	LeagueN	DivisionW	PutOuts	Assists	Errors
-Alan Ashby	475.00	315	81	7	24	38	39	14	3449	835	69	321	414	375	1	1	632	43	10
-Alvin Davis	480.00	479	130	18	66	72	76	3	1624	457	63	224	266	263	0	1	880	82	14
-Andre Dawson	500.00	496	141	20	65	78	37	11	5628	1575	225	828	838	354	1	0	200	11	3
-Andres Galarraga	91.50	321	87	10	39	42	30	2	396	101	12	48	46	33	1	0	805	40	4
-Alfredo Griffin	750.00	594	169	4	74	51	35	11	4408	1133	19	501	336	194	0	1	282	421	25
-Al Newman	70.00	185	37	1	23	8	21	2	214	42	1	30	9	24	1	0	76	127	7
-Argenis Salazar	100.00	298	73	0	24	24	7	3	509	108	0	41	37	12	0	1	121	283	9
-Andres Thomas	75.00	323	81	6	26	32	8	2	341	86	6	32	34	8	1	1	143	290	19
-Andre Thornton	1100.00	401	92	17	49	66	65	13	5206	1332	253	784	890	866	0	0	0	0	0
-Alan Trammell	517.14	574	159	21	107	75	59	10	4631	1300	90	702	504	488	0	0	238	445	22
-Alex Trevino	512.50	202	53	4	31	26	27	9	1876	467	15	192	186	161	1	1	304	45	11
-Andy VanSlyke	550.00	418	113	13	48	61	47	4	1512	392	41	205	204	203	1	0	211	11	7
-Alan Wiggins	700.00	239	60	0	30	11	22	6	1941	510	4	309	103	207	0	0	121	151	6
-Bill Almon	240.00	196	43	7	29	27	30	13	3231	825	36	376	290	238	1	0	80	45	8
-Buddy Bell	775.00	568	158	20	89	75	73	15	8068	2273	177	1045	993	732	1	1	105	290	10
-Buddy Biancalana	175.00	190	46	2	24	8	15	5	479	102	5	65	23	39	0	1	102	177	16
-Bruce Bochy	135.00	127	32	8	16	22	14	8	727	180	24	67	82	56	1	1	202	22	2
-Barry Bonds	100.00	413	92	16	72	48	65	1	413	92	16	72	48	65	1	0	280	9	5
-Bobby Bonilla	115.00	426	109	3	55	43	62	1	426	109	3	55	43	62	0	1	361	22	2
-Bob Brenly	600.00	472	116	16	60	62	74	6	1924	489	67	242	251	240	1	1	518	55	3
-Bill Buckner	776.67	629	168	18	73	102	40	18	8424	2464	164	1008	1072	402	0	0	1067	157	14
-Brett Butler	765.00	587	163	4	92	51	70	6	2695	747	17	442	198	317	0	0	434	9	3
-Bob Dernier	708.33	324	73	4	32	18	22	7	1931	491	13	291	108	180	1	0	222	3	3
-Bo Diaz	750.00	474	129	10	50	56	40	10	2331	604	61	246	327	166	1	1	732	83	13
-Bill Doran	625.00	550	152	6	92	37	81	5	2308	633	32	349	182	308	1	1	262	329	16
-Brian Downing	900.00	513	137	20	90	95	90	14	5201	1382	166	763	734	784	0	1	267	5	3
-Billy Hatcher	110.00	419	108	6	55	36	22	3	591	149	8	80	46	31	1	1	226	7	4
-Brook Jacoby	612.50	583	168	17	83	80	56	5	1646	452	44	219	208	136	0	0	109	292	25
-Bob Kearney	300.00	204	49	6	23	25	12	7	1309	308	27	126	132	66	0	1	419	46	5
-Bill Madlock	850.00	379	106	10	38	60	30	14	6207	1906	146	859	803	571	1	1	72	170	24
-Bob Melvin	90.00	268	60	5	24	25	15	2	350	78	5	34	29	18	1	1	442	59	6
-BillyJo Robidoux	67.50	181	41	1	15	21	33	2	232	50	4	20	29	45	0	0	326	29	5
-Bill Schroeder	180.00	217	46	7	32	19	9	4	694	160	32	86	76	32	0	0	307	25	1
-Chris Bando	305.00	254	68	2	28	26	22	6	999	236	21	108	117	118	0	0	359	30	4
-Chris Brown	215.00	416	132	7	57	49	33	3	932	273	24	113	121	80	1	1	73	177	18
-Carmen Castillo	247.50	205	57	8	34	32	9	5	756	192	32	117	107	51	0	0	58	4	4
-Chili Davis	815.00	526	146	13	71	70	84	6	2648	715	77	352	342	289	1	1	303	9	9
-Carlton Fisk	875.00	457	101	14	42	63	22	17	6521	1767	281	1003	977	619	0	1	389	39	4
-Curt Ford	70.00	214	53	2	30	29	23	2	226	59	2	32	32	27	1	0	109	7	3
-Carney Lansford	1200.00	591	168	19	80	72	39	9	4478	1307	113	634	563	319	0	1	67	147	4

Exemple


Salary
AtBat
Hits
HmRun
Runs
RBI
Walks
Years
CAtBat
CHits
CHmRun
CRuns
CRBI
CWalks
LeagueN
DivisionW
PutOuts
Assists
Errors
NewLeagueN
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

Modèle de régression linéaire

Modèle de régression linéaire : $x_i = (x_i^1, \ldots, x_i^p)'$ On suppose que :

$$Y_i = \beta_0 + \beta_1 x_i^1 + \ldots + \beta_p x_i^p + \epsilon_i, 1 \leq i \leq n$$

où

- $\beta_0, \beta_1, ..., \beta_p$ sont les paramètres du modèle (à estimer),
- les variables ε_i vérifient : $E[\varepsilon_i] = 0$, $cov(\varepsilon_i, \varepsilon_j) = 0 \ \forall i \neq j$, $var(\varepsilon_i) = \sigma^2$.
- les variables sont supposées gaussiennes pour l'inférence statistique (intervalles de confiance, tests, p valeurs...)

Modèle de régression linéaire sous forme matricielle :

$$Y = \mathbb{X}\beta + \varepsilon$$

où

$$\mathbb{X} = \begin{pmatrix} x_1^0 & x_1^1 & x_1^2 & . & x_1^p \\ x_2^0 & x_2^1 & x_2^2 & . & x_2^p \\ . & . & . & . & . \\ x_n^0 & x_n^1 & x_n^2 & . & x_n^p \end{pmatrix}, \ \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ . \\ . \\ \beta_p \end{pmatrix}, \ \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ . \\ . \\ \varepsilon_n \end{pmatrix},$$

$$x^0 = \begin{pmatrix} x_1^0 \\ x_2^0 \\ . \\ . \\ x_n^0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ . \\ . \\ 1 \end{pmatrix}, \ x^j = \begin{pmatrix} x_1^j \\ x_2^j \\ . \\ . \\ x_n^j \end{pmatrix} \ (j = 1 \dots p).$$

Rappel : Modèle de régression linéaire

L'estimateur des Moindres Carrés Ordinaires (MCO) $\hat{\beta}$ de β minimise la quantité :

$$\sum_{i=1}^{n}\left(y_i-\sum_{j=1}^{p}\beta_i x_{ij}\right)^2 =\parallel Y-X\beta \parallel^2=(y-X\beta)'(y-X\beta)$$

Que l'on peut récrire

$$\widehat{\beta}=\underset{\beta}{Min}\sum_{i=1}^{n}\left(y_i-\sum_{j=1}^{p}\beta_i x_{ij}\right)^2=\underset{\beta}{Min}\parallel Y-X\beta \parallel^2=\underset{\beta}{Min}(y-X\beta)'(y-X\beta)$$

Estimateurs par la méthode MCO

$$\underset{\beta_{\mathbf{0}},\ldots,\beta_p}{Min}\{S(\beta_0,\ldots,\beta_p)=\sum_{i=1}^{n}(Y_i-(\beta_0+\beta_1 x_i^1+\ldots+\beta_1 x_i^p))^2\}$$

Cet minimum est atteint tel que :

$$\begin{cases}\frac{\partial S}{\partial \hat{\beta}_{\mathbf{0}}}=-2\sum_{i=1}^{n}(y_i-\hat{\beta}_0-\hat{\beta}_1 x_i^1-\cdots-\hat{\beta}_p x_i^p)1=0 & (1)\\ \frac{\partial S}{\partial \hat{\beta}_{\mathbf{1}}}=-2\sum_{i=1}^{n}(y_i-\hat{\beta}_0-\hat{\beta}_1 x_i^p-\cdots-\hat{\beta}_p x_i^p)x_i^1=0 & (2)\\ \vdots & \\ \frac{\partial S}{\partial \hat{\beta}_p}=-2\sum_{i=1}^{n}(y_i-\hat{\beta}_1-\hat{\beta}_0 x_i^1-\cdots-\hat{\beta}_p x_i^p)x_i^p=0 & (p+1)\end{cases}$$

Estimateurs par la méthode MCO

La solution du système des équations fournit les estimateurs des paramètres $\beta_0, \ldots, \beta_p$ que l'on note $\widehat{\beta}_0, \ldots, \widehat{\beta}_p$

$$\hat{\beta} = (X'X)^{-1}X'y$$

- Vecteur des valeurs ajustées : $\hat{Y} = X\hat{\beta} = X(X'X)^{-1}X'Y$
- Vecteur des résidus : $\hat{\varepsilon} = Y - \hat{Y}$
- Somme des carrés résiduelle : $SCR = \sum_{i=1}^{n} \varepsilon_i^2$
- Somme des carrés totale : $SCT = \sum_{i=1}^{n} (Y_i - \bar{Y})^2$
- Somme des carrés expliquée : $SCE = \sum_{i=1}^{n} (\hat{Y}_i - \bar{Y})^2$
- Équation d'analyse de la variance : $SCT = SCE + SCR$
- Coefficient de détermination : $R^2 = 1 - SCR/SC$

Exemple : Coefficients de la regression linéaire par MCO

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  163.10359   90.77854   1.797 0.073622 .
xAtBat        -1.97987    0.63398  -3.123 0.002008 **
xHits          7.50077    2.37753   3.155 0.001808 **
xHmRun         4.33088    6.20145   0.698 0.485616
xRuns         -2.37621    2.98076  -0.797 0.426122
xRBI          -1.04496    2.60088  -0.402 0.688204
xWalks         6.23129    1.82850   3.408 0.000766 ***
xYears        -3.48905   12.41219  -0.281 0.778874
xCAtBat       -0.17134    0.13524  -1.267 0.206380
xCHits         0.13399    0.67455   0.199 0.842713
xCHmRun       -0.17286    1.61724  -0.107 0.914967
xCRuns         1.45430    0.75046   1.938 0.053795 .
xCRBI          0.80771    0.69262   1.166 0.244691
xCWalks       -0.81157    0.32808  -2.474 0.014057 *
xLeagueN      62.59942   79.26140   0.790 0.430424
xDivisionW  -116.84925   40.36695  -2.895 0.004141 **
xPutOuts       0.28189    0.07744   3.640 0.000333 ***
xAssists       0.37107    0.22120   1.678 0.094723 .
xErrors       -3.36076    4.39163  -0.765 0.444857
xNewLeagueN  -24.76233   79.00263  -0.313 0.754218
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Problèmes de la régression

$$\text{Variance de l'estimateur des MCO } V(\hat{\beta}_j) = \frac{\hat{\sigma}_\varepsilon}{n}\nu_j \text{ Avec } \nu_j = \frac{1}{1 - R_j^2}$$

R_j^2 est le coefficient de détermination de la régression de x^j sur les (p-1) autres variables.
Problème de Colinéarité : $R_j^2 \approx 1 \Rightarrow \nu_j \approx \infty$

$\hat{\sigma}_\varepsilon = \frac{1}{n-p}\sum_i \hat{\varepsilon}_i$ Variance estimée de l'erreur : SCR (somme des carrés des résidus) : indicateur de qualité de la régression, divisé par les degrés de liberté.

Problème de Dimensionnalité : $p \approx n \Rightarrow \hat{\sigma}_\varepsilon \approx \infty$ et ; $p > n \Rightarrow X'X$ n'est pas inversible

Conséquence : 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.

Idée : Rechercher des estimateurs biaisées avec une variance petite

Compromis biais variance

Il existe un compromis entre la capacité d'un modèle à minimiser le biais et la variance.

- Importance capitale : construire un modele parcimonieux :
  - Reduire le **nombre de variables explicatives**
  - Reduire le Nombre de feuilles dans un arbre de classification
  - Reduire le Nombre de couches cachees dans un réseau de Neurones
- Modele complexe $\Longrightarrow$ bon ajustement aux donnees (Ajouter des variables en regression réduit le biais) donc **Erreur d'ajustement faible.** MAIS **Ajouter des variables augmente la variance**
- Modele réduit $\Longrightarrow$ Faible variance du modele. MAIS mauvaise qualité d'ajustement

Objectif : optimiser un dosage entre biais et variance en controlant l'ajustement aux donées, et la complexité du modèle

Compromis biais variance

- On minimise en f, à la vue des observations x, ce qui justifie :
- On recherche f, fonction de x qui minimise : $\mathbb{E}\left[(y - f(x))^2 \mid x\right]$
- Proposition : Parmi toutes les fonctions de x, la solution qui minimise (1) est donnée par $f(x) = \mathbb{E}[y \mid x]$
- Proposition : Connaissant $\mathcal{D} = \{(X_1, Y_1), \ldots (X_n, Y_n)\}$, on a le compromis biais/variance :

$$\mathbb{E}\left[(y - f(x, \mathcal{D})^2 \mid \mathcal{D}\right] = \mathbb{E}\left[(y - \mathbb{E}[y \mid x])^2\right] + \underbrace{\mathbb{E}_{\mathcal{D}}\left[(f(x, \mathcal{D}) - \mathbb{E}[y \mid x])^2\right]}_{\text{Biais}}$$
$$+ \underbrace{\mathbb{E}_{\mathcal{D}}\left[(f(x, \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(x, \mathcal{D})])^2\right]}_{\text{Variance}}$$

- Problème : Loi de y en général inconnue, $\mathbb{E}[y \mid x]$ souvent incalculable. Approche "optimale" pas touiours réalisable.

Le compromis biais variance en Regression

Soit x_{n+1} supplémentaire
$\hat{y}_{n+1} = \hat{\beta}_0 + \hat{\beta}_1 x^1_{n+1} + \ldots + \hat{\beta}_p x^p_{n+1}$

La qualité de la prédiction est évalué l'aide de :

$$E[(y_{n+1} - \hat{y}_{n+1})^2] = \sigma^2 + (E(\hat{y}_{n+1}) - y_{n+1})^2 + E[((\hat{y}_{n+1} - E(\hat{y}_{n+1}))^2]$$

σ^2 : Erreur incompressible. Variance de la cible Y, on ne pourra jamais faire mieux.

$(E(\hat{y}_{n+1}) - y_{n+1})^2$: $Biais^2$. 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.).

$E[((\hat{y}_{n+1} - E(\hat{y}_{n+1}))^2]$ 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.

Principe de la régularisation

$$E[(y_{n+1} - \hat{y}_{n+1})^2] = \sigma^2 + (E(\hat{y}_{n+1}) - y_{n+1})^2 + E[((\hat{y}_{n+1} - E(\hat{y}_{n+1}))^2]$$

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 $\hat{\beta}_j$ 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

RIDGE

RÉGRESSION RIDGE

Contrainte sur la norme L2 des coefficients

$$\underset{\beta_1,\ldots,\beta_p}{min} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_i^j)^2 \text{ Sous la contrainte } \sum_{j+1}^{p}\beta_j^2 \leq \tau,\ \tau \geq 0$$

ou de manière totalement équivalente :

$$\underset{\beta_1,\ldots,\beta_p}{min} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_i^j)^2 + \lambda \sum_{j=1}^{p}\beta_j^2$$

Le second terme est une fonction de pénalité
$\lambda(\lambda \geq 0)$ est un paramètre (coefficient de pénalité) qui permet de contrôler l'impact de la pénalité : **à fixer**

$$\underset{\beta}{min}(\mathrm{y} - \mathrm{X}\beta)'(\mathrm{y} - \mathrm{X}\beta) + \lambda\,(\beta'\beta)$$

$$\underset{\beta}{min}\mathrm{y}'\mathrm{y} + \beta'\mathrm{X}'\mathrm{X}\beta - 2\mathrm{y}'\mathrm{X}\beta + \lambda\,(\beta'\beta)$$

$$\frac{\partial}{\partial\beta} = 2\mathrm{X}'\mathrm{X}\beta - 2\mathrm{X}'\mathrm{y} + 2\lambda\beta = (\mathrm{X}'\mathrm{X} + \lambda\mathrm{I})\,\beta - \mathrm{X}'\mathrm{y} = 0$$

Solution

$$\beta_{R(\lambda)} = \left(\mathrm{X}'\mathrm{X} + \lambda\mathrm{I}\right)^{-1}\mathrm{X}'\mathrm{y}$$

Quelques remarques sur la Régression RIDGE

- X est standardisée et le vecteur Y est centré
- L'inclusion de λ rend le problème non singulier, même si la matrice $X'X$ n'est pas inversible.
- Si deux prédicteurs sont fortement corrélés, leur coefficient sont similaires.
  - Utile si le but est la sélection de groupe de variables exogènes redondantes
  - Pas utile pour faire de la sélection d'une variable dans un groupe de variables redondantes

Propriétés des coefficients RIDGE

Supposons la décomposition en valeurs singulière de la matrice X des prédciteurs

$$X = UDV'$$

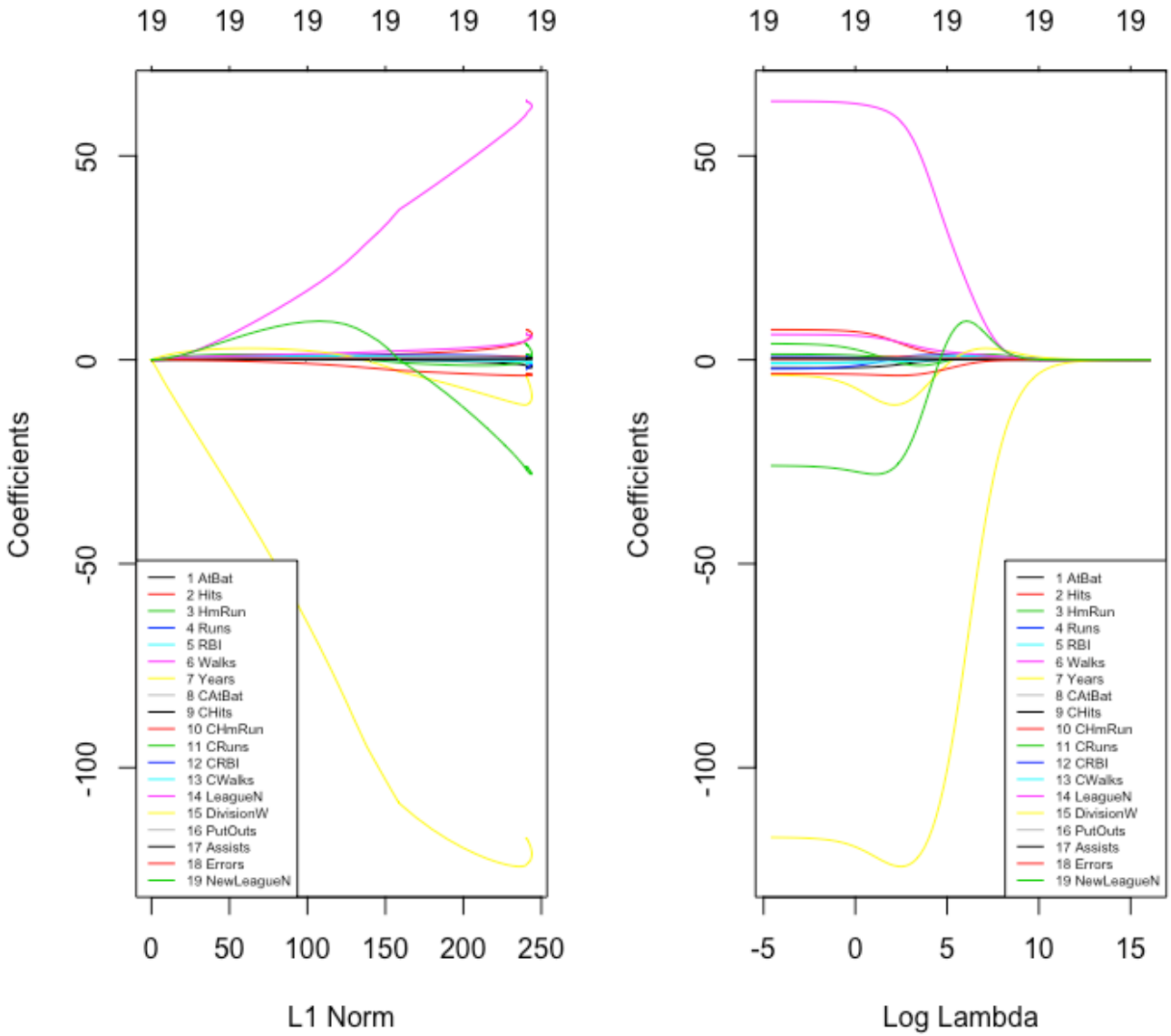
où

- $D(p \times p)$ est une matrice diagonale des valeurs singulières de X $d_1 < d_2 < d_3 ...$
- U est la matrice orthogonale $(N \times P)$ des vecteurs singuliers de gauche de X
- V est la matrice orthogonale $(P \times P)$ des vecteurs singuliers de droite de X

On démontre que

$$\begin{aligned} \hat{\beta}_{RIDGE} &= (X'X + \lambda I)^{-1} X'y \\ &= Vdiag(\frac{d_p}{d_p^2 + \lambda}) U'y \ 1<p<P \end{aligned}$$

Exemple : Hitters



RÉGRESSION RIDGE

Détermination de la valeur de λ

S'appuyer sur un critère de performance pure

$$ERR = \frac{1}{\sharp D_{test}} \sum_{i \in D_{test}} (y_i - \hat{y}_i)^2$$

Principe

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

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

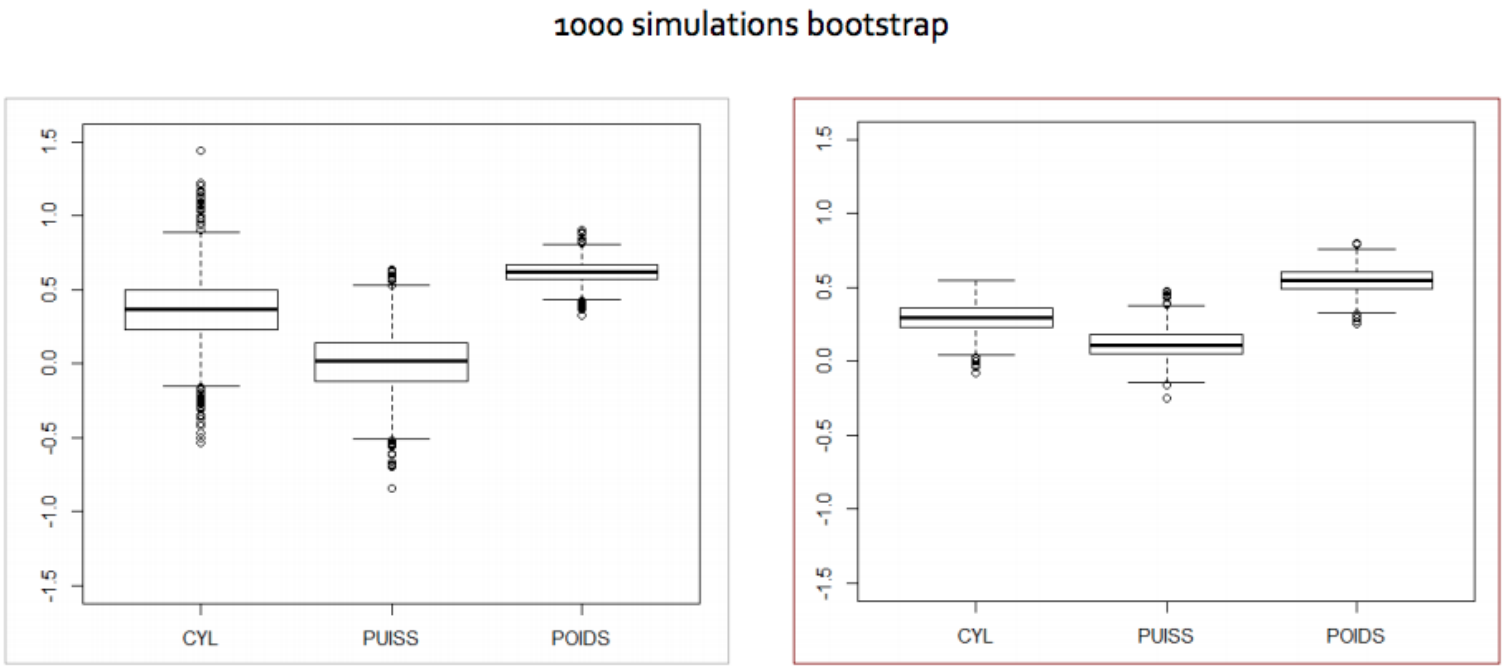
Dans le cas particulier où K = n pour la validation croisée, nous avons le schéma leave-one-out (LOOCV)

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

Qui peut être approcher sans avoir à construire explicitement les (n-1) modèles par

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

RÉGRESSION RIDGE : Exemple « CARS »



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

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

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

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

Figure – Source : Ricco

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.

RÉGRESSION LASSO

Contrainte sur la norme L1 des coefficients
Très similaire à Ridge

$$\text{Critère : } \underset{\beta_0,\ldots,\beta_1}{min} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 \text{ Sous contrainte } \sum_{j=1}^{p}|\beta_j| \leq \tau$$

Avec la fonction de pénalité

$$\underset{\beta_0,\ldots,\beta_1}{min} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j|$$

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

Il n'y a pas de calcul direct pour LASSO.

Algorithm 3.2 *Least Angle Regression.*

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.
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.

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.

Figure – Source : Ricco

LASSO Path – Exemple ''CARS''

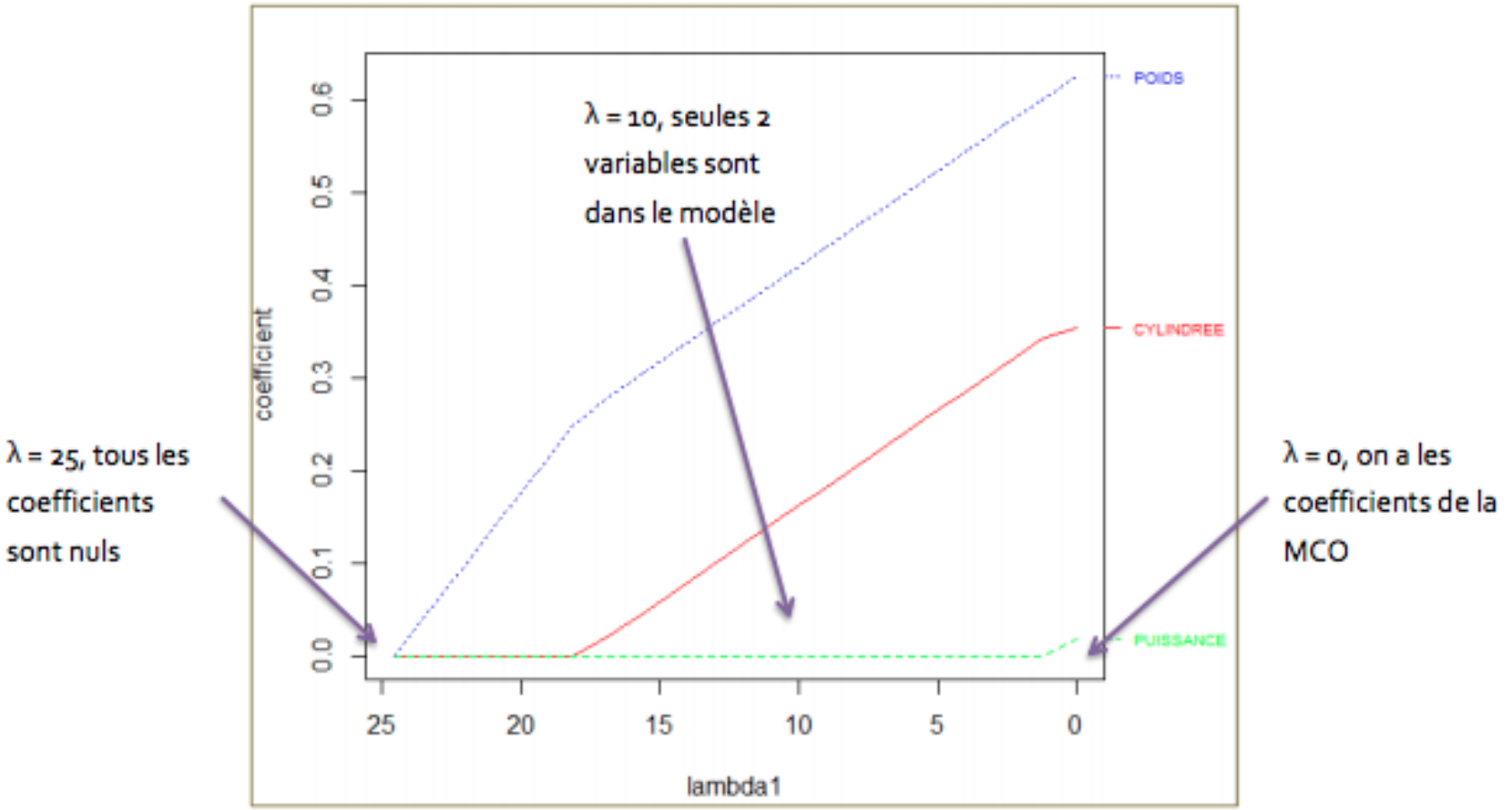


Figure – Source : Ricco

Choix du λ par VC

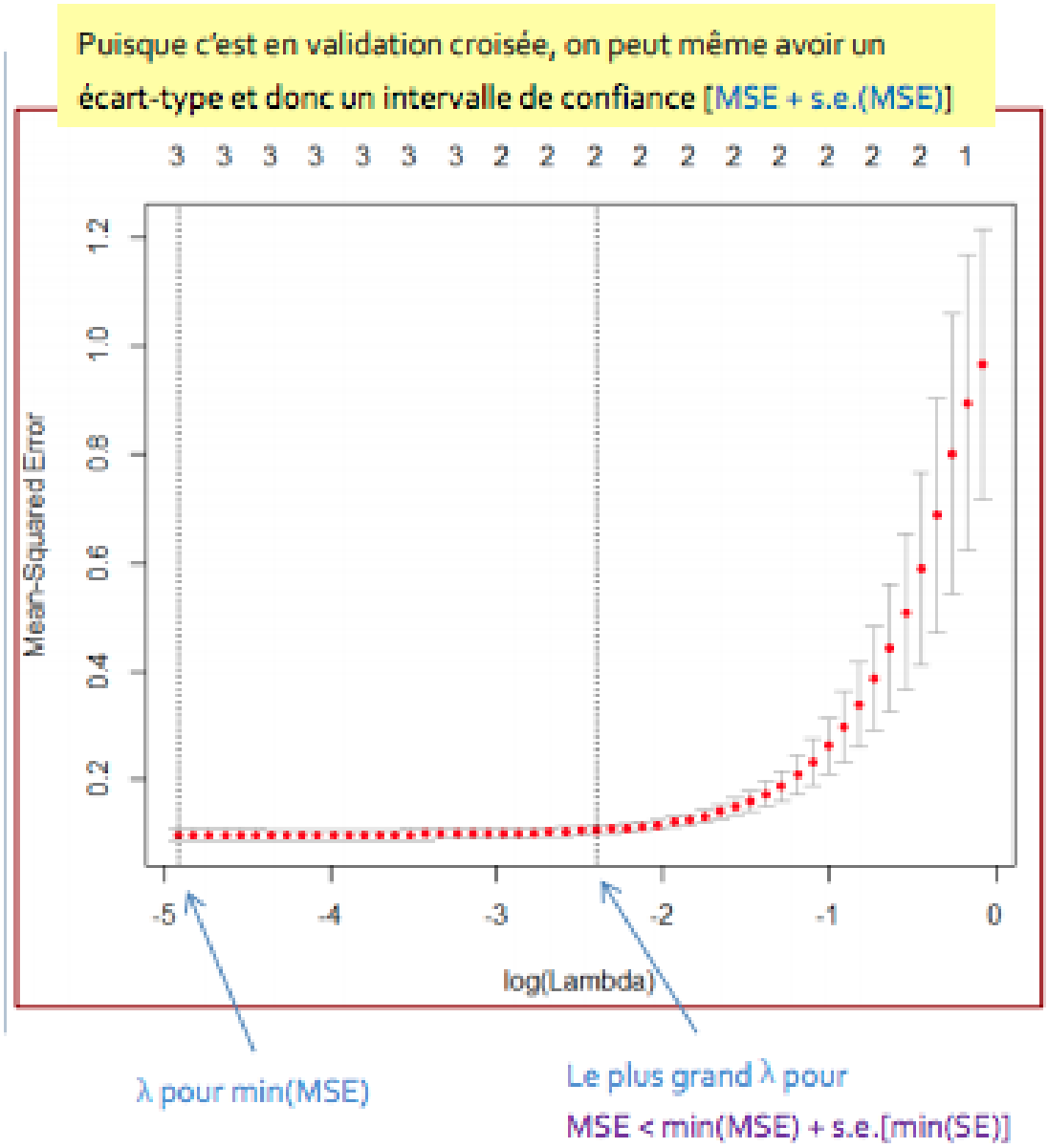


Figure – Source : Ricco

Bilan LASSO

Avanatge : Capacité à sélectionner les variables en acceptant les coefficients nuls

Inconvénients

- Dans les problèmes à très grandes dimensions ($p >> n$), 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 a cible souvent, masquant l'influence des autres. Cet inconvénient est inhérent aux techniques intégrant un mécanisme de sélection de variables

06979073

ELASTIC NETS : Mix de RIDGE et LASSO

Combiner les avantages de Ridge et Lasso

Formulation sous la forme d'un problème d'optimisation

$$\min_{\beta_0,\ldots,\beta_1} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda_1 \sum_{j=1}^{p}|\beta_j| + \lambda_2 \sum_{j=1}^{p}\beta_j l^2$$

$\lambda_1 \geq 0$ et $\lambda_2 \geq 0$; ($\lambda_1 = 0$ et $\lambda_2 > 0$) : Ridge ;Si ($\lambda_1 > 0$ et $\lambda_2 = 0$) : Lasso ;

Si ($\lambda_1 = 0$ et $\lambda_2 = 0$) : MCO.

Formulation alternative de l'optimisation

$$\min_{\beta_0,\ldots,\beta_1} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda(\alpha \sum_{j=1}^{p}|\beta_j| + (1-\alpha)\sum_{j=1}^{p}\beta_j^2$$

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

ELASTIC NETS : Mix de RIDGE et LASSO

- C'est un compromis entre les régressions Ridge et LASSO
- La partie quadratique de la pénalité :
  - Supprime la limitation de LASSO sur le nombre de variables sélectionnées ;
  - Stabilise le chemin de régularisation $\mathbf{L_1}$
  - Encourage la sélection du groupe
- Deux paramètres à estimer : α et λ
  - $-\alpha = 0$-> Régression Ridge
  - $-\alpha = 1- >$ Régression LASSO
- Des méthodes numériques sont utilisés pour calculer des estimations Pas

Quel intérêt ?

- 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

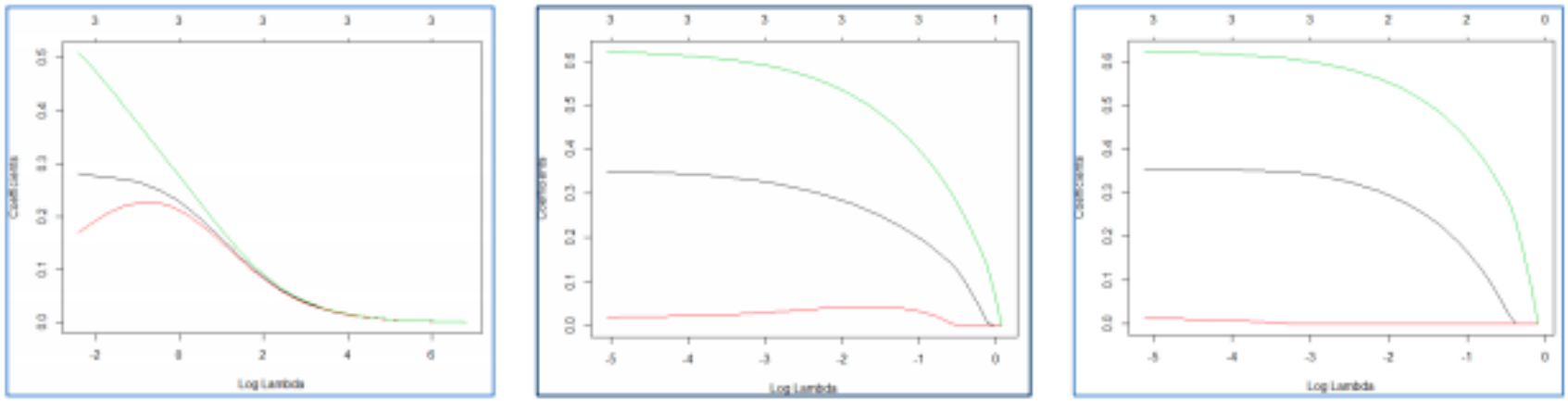


Figure – Source : Ricco

Des méthodes numériques sont utilisés pour calculer des estimations Pas de prédiction invariante par rapport au changement d'échelle (X standardisée, y centrée)

Si beaucoup d'observations

1. On partitionne au hasard les observations dans 2 ou 3 sous-échantillons : Apprentissage, Validation et Test (disons 50%, 25% et 25% des obs)

2. Sélection du modèle :

    1. On lance tous les modèles (pour toutes les valeurs «possibles» du paramètre de régularisation lambda) sur le set d'apprentissage

    2. On évalue tous les modèles du jeu de validation et je choisis le meilleur modèle, typiquement à l'aide de écart type moyen de prédiction (MSEP), calculé comme la moyenne des carrés des erreurs de prédiction

    3. Le meilleur modèle est celui avec le moindre $\mathrm{MSEP}_{\mathbf{val}}$.

    4. . Evaluation du modèle : on calcule le MSEP $_{\mathbf{test}}$ sur l'échantillon de test

    5. On lance le modèle sur tout l'échantillon et on obtiens les coefficients finals

Selection du modèle par VC

1. On partitionne au hasard mes observations dans 2 sous-échantillons : Apprentissage (disons le 75% des obs) et Test.
2. Sélection du modèle : Pour chaque valeur possible du paramètre de régularisation :
   1. Les observations sont divisées en G groupes
   2. L'algorithme est exécuté G fois sur les observations n'appartenant pas au groupe g
   3. Chaque fois, on calcule la valeur prédite de y pour les observations appartenant au groupe g
   4. On estime le MSEP $_{\text{cv}}$ par la moyenne du MSEP pour chaque groupe est on choisie le meilleur modèle
   5. On estime l'écart type du MSEP $_{\text{cv}}$ par l'écart type du MSEP moyen de chaque groupe

   1) 3. Evaluation du modèle : on calcule le MSEP $_{\text{test}}$ sur l'échantillon de test 4. On lance le modèle sur tout l'échantillon et on obtient les coefficients finals

Regression sur Composantes Prinicpales
PCR

Regression sur composantes principales (PCR) : Démarche

1. Rechercher Q composantes orthogonales $c_q = Xv_q$ (On générale Q s'obtient par validation croisée).
2. Régresser y sur les composantes c_q
3. Exprimer l'équation de la régression en fonction de X

PCR : Equation de la régression

L'information rédondante dans X est résumée dans les Q composantes principales

$$C_Q = [c_1, \ldots, c_q \ldots c_Q] = XV_Q = U_Q D_Q$$

$$\begin{aligned}
\hat{Y}_{PCR} &= C_Q(C_Q C_Q')^{-1} C_Q' Y \\
&= XV_Q((XV_Q)'(XV_Q))^{-1}(XV_Q)' Y \\
&= XV_Q(V_Q' X' XV_Q)^{-1} V_Q X' Y \\
&= XV_Q(D_Q^2)^{-1} V_Q X' Y \\
&= (XV_Q D_Q^{-1})(D_Q^{-1} V_Q X') Y \\
&= U_Q U_Q' Y
\end{aligned}$$

où

- D_Q^2 est la matrice diagonale composée des Q plus grandes valeurs propres de $X'X$
- V_Q^2 est la matrice des Q vecteurs propres de $X'X$
- U_Q^2 est la matrice des Q vecteurs propres de XX'

PCR les +/-

- Avantages
  - Gère les jeux de données "Larges" (p>n)
  - Diminue la variabilité des estimateurs en raison de multicolinearité
  - Outils de visualisation
- Inconvénients
  - Estimateurs biaisés
  - Solution non équivalente par rapport au changement d'échelle.
  - Les composantes sont dépendantes de la structure de corrélation de X sans tenir compte de la corrélation entre Y et les prédicteurs

Régression PLS

Régréssion des moindres carrés partiels (PLS-R)

- PLS-R est une technique de régression qui permet de lier ensemble de variables dépendantes à une ou plusieurs variables de réponses
- PLS-R décompose la matrice des prédicteurs par une extraction séquentielle des composantes orthogonales qui résument les variables tout en prévoyant les réponses.

Critère de la PLS

Pour $q = 1, \ldots, Q$ nous recherchons des composante $t_q = Xa_q$ qui maximisent le critère suivant :

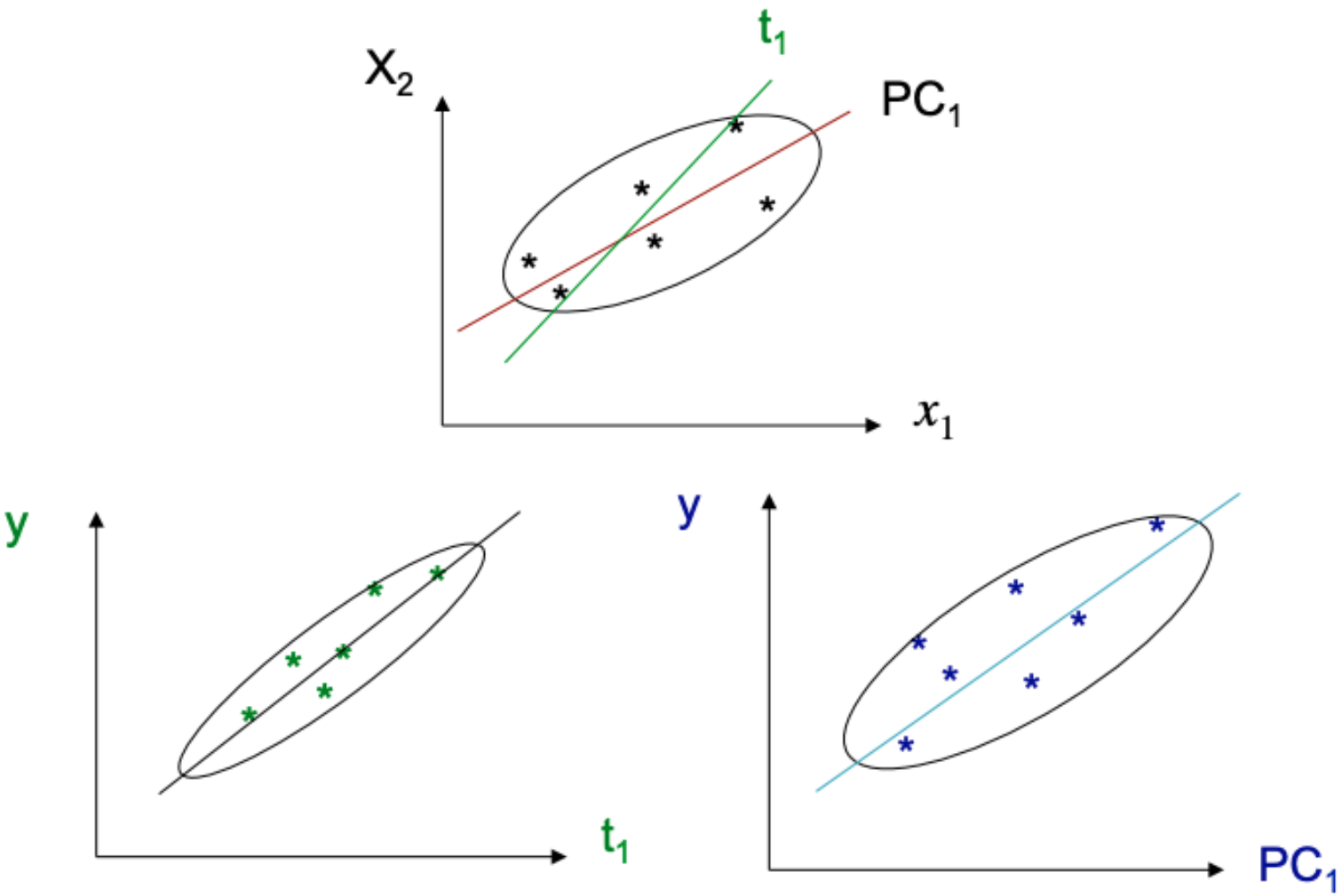
$$\underset{\|a_q\|=1}{argmax}\{cov^2(Xa_h, y)\}$$

Soumis à une contraintes de normalisation pour a_q et d'orthogonalisation pour les t_q

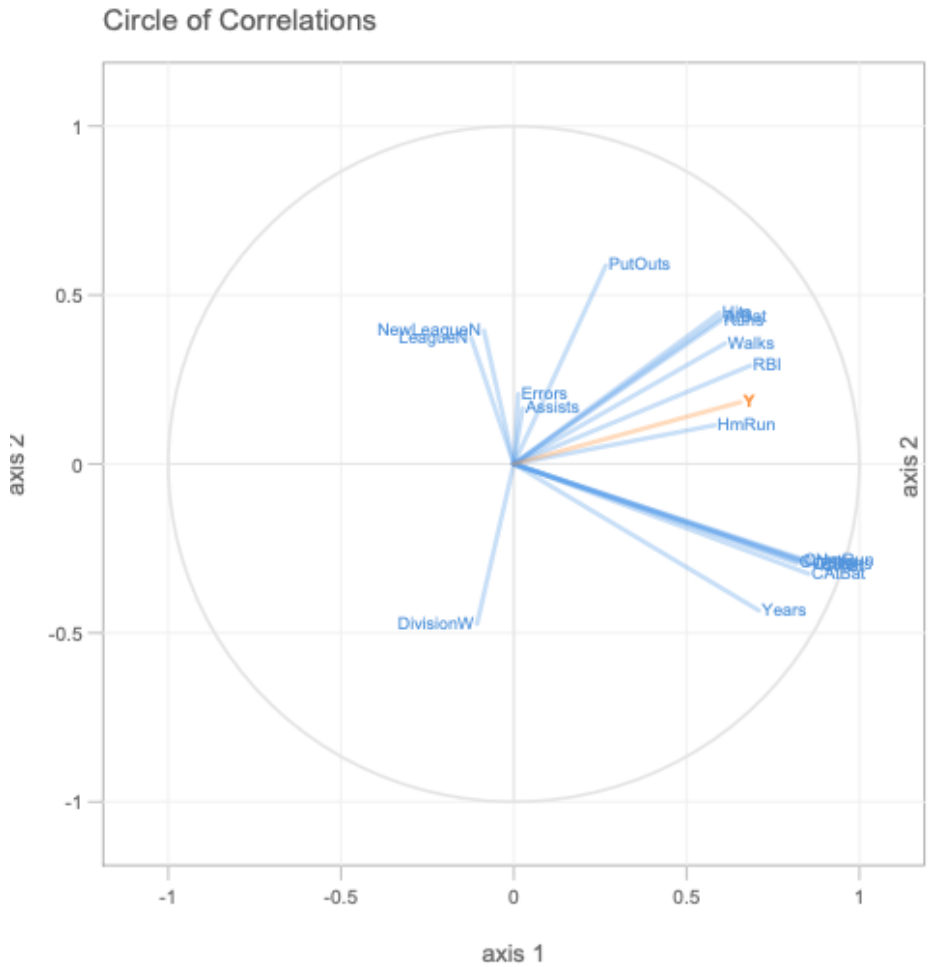
La PLS-R aboutit à un compromis entre la régression linéaire multiple de y sur X et l'analyse en composante principale de X. En effet :

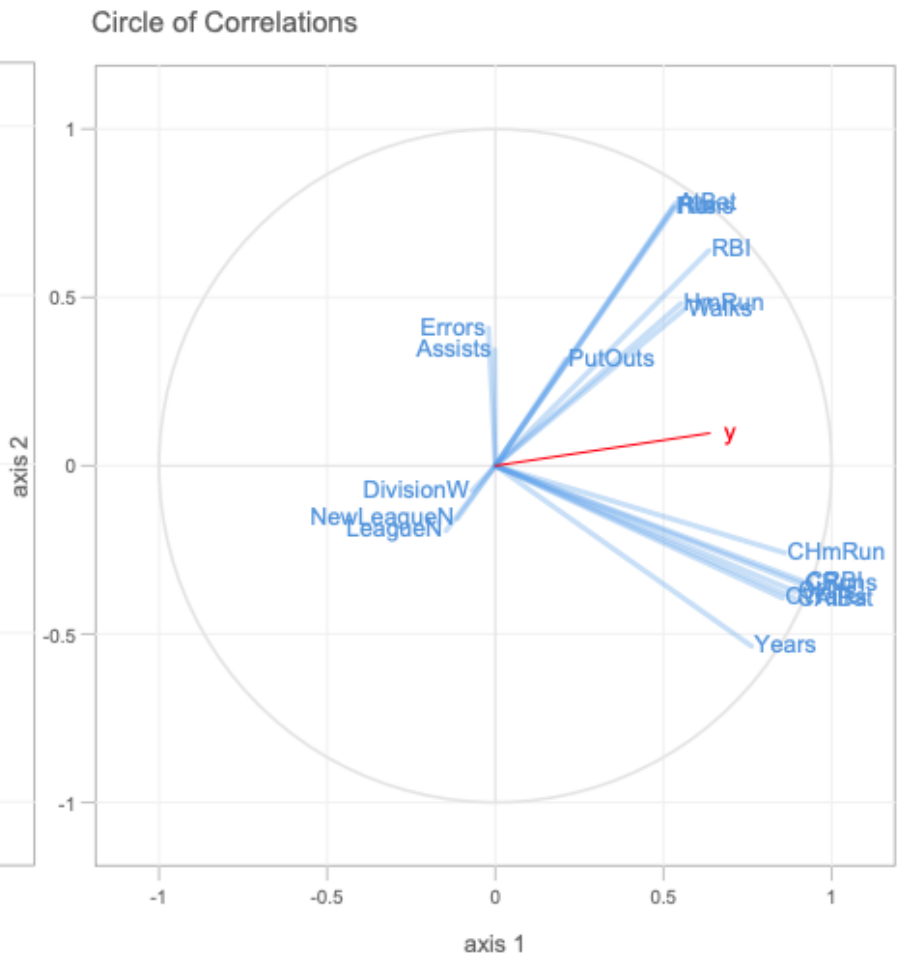
$$cov^2(t_q, y) = cor^2(t_q, y) * var(t_h) * var(y)$$

Interprétation géométrique



Cercle des corrélations : PLS-R Vs PCR





PLS-R : Démarche

1. Rechercher Q composantes orthogonales $t_q = Xa_q$ (On générale Q s'obtient par validation croisée).
2. Régresser y sur les composantes PLS t_q
3. Exprimer l'équation de la régression en fonction de X

Détermination de la première composante PLS t_1

1. Calculer la première composante :

$$t_1 = Xa_1 = \sum_{j=1}^{P} cor(y, X_j)X_j$$

2. Normalisation du vecteur $a_1 = (a_{11}, \ldots, a_{1k})$

3. Calcul des résidus ε_1 et ξ_1 des régressions de y et X sur t_1 :
   - Régresser y sur t_1 :
   $$y = \beta_1 t_1 + \varepsilon_1$$
   - Régresser X sur t_1 :
   $$X = t_1 \gamma_1' + \xi_1$$

Détermination de la deuxième composante PLS t_2

1. Calculer la deuxième composante :

$$t_2 = \xi_1 b_2 = \sum_{j=1}^{P} cor(\varepsilon_1, X_{1j}) X_{1j}$$

2. Normalisation du vecteur $b_2 = (b_{21}, \ldots, b_{2Q})$
3. Calcul de a_2 tel que : $t_2 = \xi_{1b_2} = Xa_2$
4. Régresser ε_1 sur $t_2 - Xa_2$:

$$y_1 = \beta_2 t_2 + \varepsilon_2$$

5. Calcul des résidus ε_2 et ξ_2 des régressions de y et ξ_1 sur t_2 :
   - Régresser y sur t_1 :

$$\varepsilon_1 = \beta_2 t_2 + \varepsilon_2$$

   - Régresser X sur t_1 :

$$\xi_1 = t_2 \gamma_2' + \xi_2$$

Le processus continue pour les composantes d'ordre supérieures.

Régression de la variable y sur les composants PLS

$$\hat{y}_{PLS} = T_Q(T_Q' T_Q)^{-1} T_Q' y$$

Chaque composante t_q peut s'exprimer comme fonction de X :

$$t_Q = X_{Q-1} W_Q = X w_Q^* = w_{Q_1}^* X_1 + \ldots + w_{Q_p}^* X_p$$

où

$$W_Q^* = [w_1^*, \ldots, w_Q^*] = W_Q(P_Q W_Q')^{-1}$$

L'équation de régression d'un modèle à Q composantes :

$$\begin{aligned}
\hat{Y} &= c_1 t_1 + c_2 t_2 + \ldots + c_Q t_Q + \varepsilon_Q \\
&= c_1 X a_1 + c_2 X_1 a_2 + \ldots + c_Q X_{Q-1} W_Q + \varepsilon_Q \\
&= c_1 X a_1 + c_2 X_2^* + \ldots + c_Q X W_Q^* + \varepsilon_Q \\
&= X(c_1 W_1 + c_2 W_2^* + \ldots + c_Q W_Q^*) + \varepsilon_Q \\
&= X \beta_Q^{pls} + \varepsilon_Q = \hat{\beta}_1^{pls} X_1 + \hat{\beta}_2^{pls} X_2 + \ldots + \hat{\beta}_p^{pls} X_p + \varepsilon_Q
\end{aligned}$$

Propriété de shrinkage (rétrécissement) de la régression PLS

De Jong (1995) a montré que :

- La longueur de l'estimateur PLS augmente avec le nombre de composantes
- $$\| \beta_1^{pls} \| < \| \beta_2^{pls} \| < \ldots < \| \beta_{rang(X)}^{pls} \|$$

Si $Rang(X) = Q$ alors $\beta^{pls} = \beta^{MCO}$

PLS-R les +/-

- Avantages
  - Gère les jeux de données "Larges" (p>n)
  - Diminue la variabilité des estimateurs en raison de multicolinearité
  - Traite les données manquantes sans imputation a priori
  - Outils de visualisation
  - faible complexité computationnelle
- Inconvénients
  - Estimateurs biaisés
  - Solution non équivalente par rapport au changement d'échelle.
  - Les composantes sont dépendantes de la structure de corrélation de X sans tenir compte de la corrélation entre Y et les prédicteurs

Qu'est-ce qu'un arbre de décision ?

Prêt bancaire

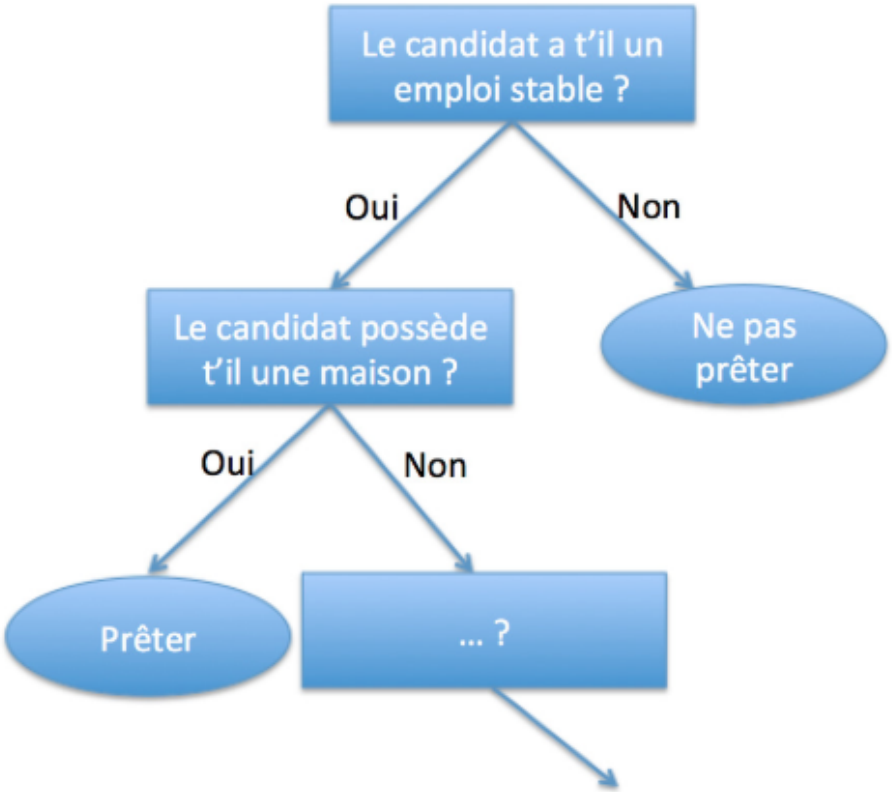


Figure – *

Source : https ://maximilienandile.github.io

Publication d'un document sur internet

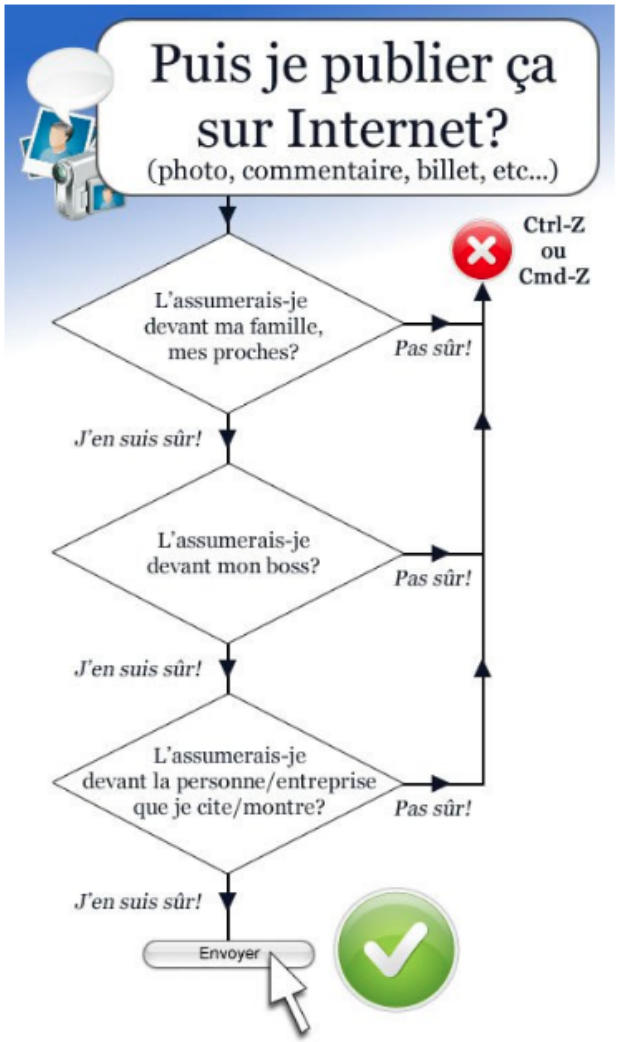


Figure – *

Source : http ://jeromechoain.files.wordpress.com

Survie des passagers sur le Titanic

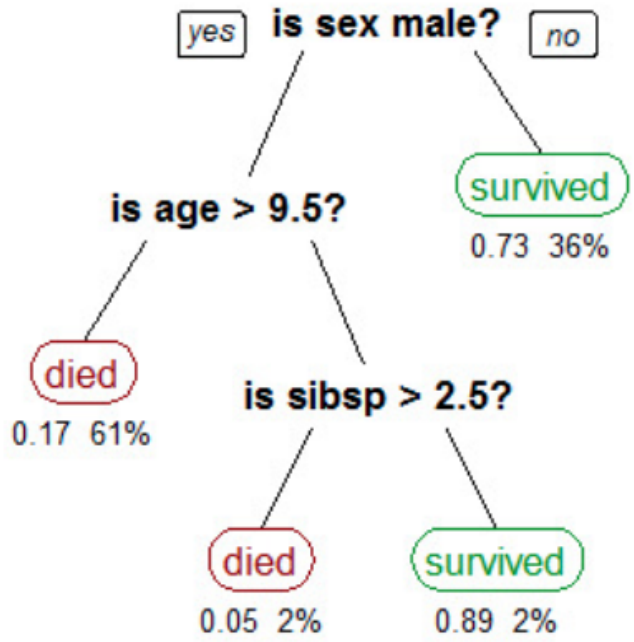
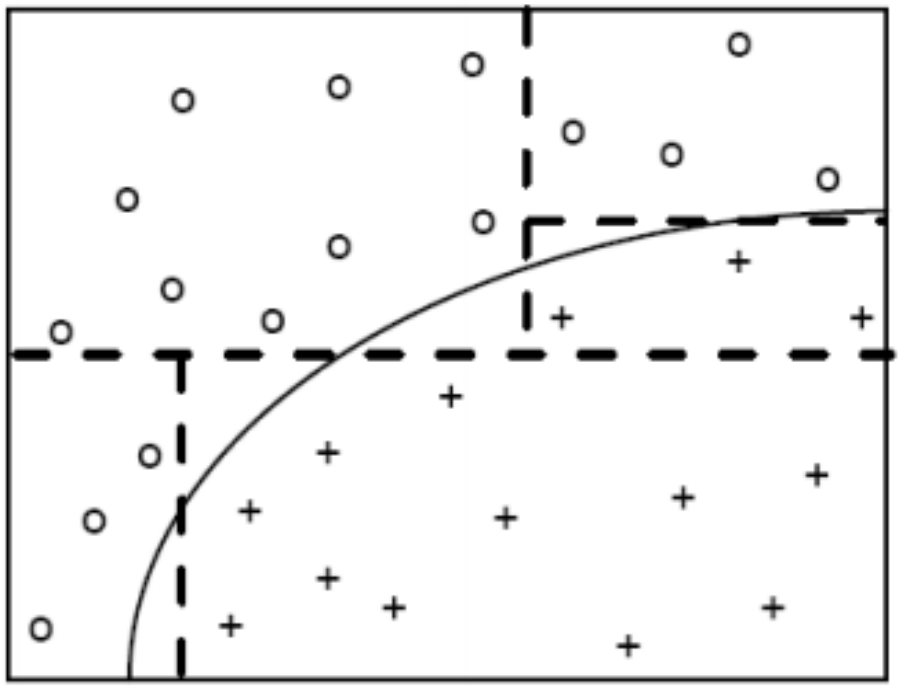


Figure – *

Source : https ://en.wikipedia.org

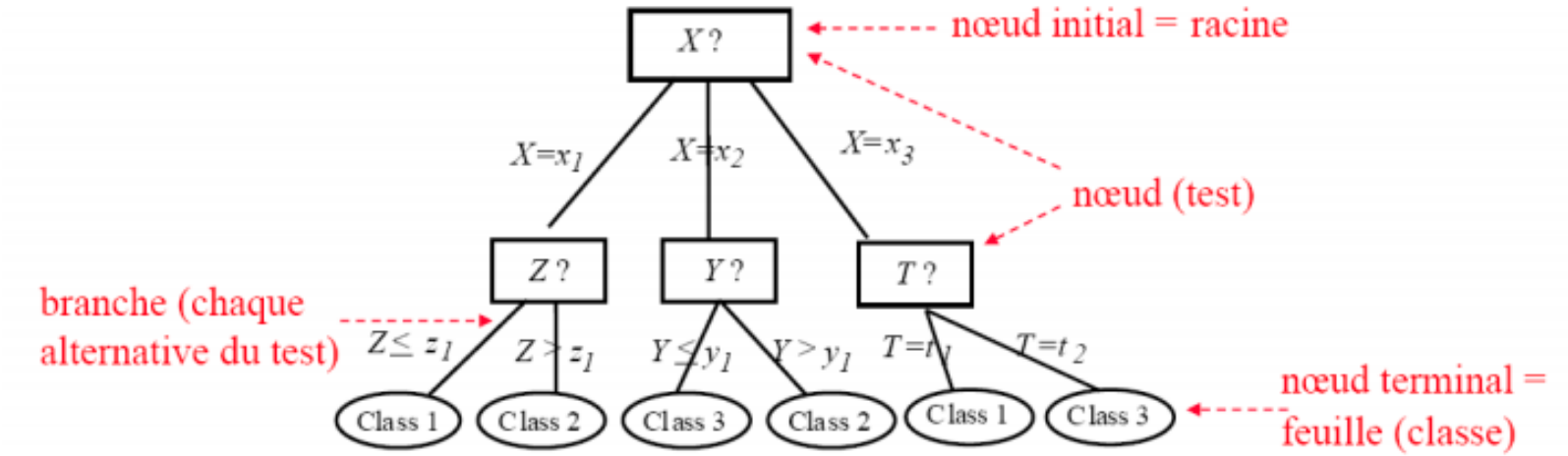
Principe général des arbres de décision

Décomposition du problème de classification en une suite de tests correspondant à ***une partition de l'espace des données en sous-régions homogènes en terme de classe***



Principe général des arbres de décision

- **Règle de classification** : aller de la racine à une feuille en effectuant les tests des nœuds
- **Classe d'une feuille** : classe majoritaire parmi les exemples d'apprentissage appartenant à cette feuille



Induction de l'arbre à partir d'une **base d'exemples d'apprentissage**

Recherche exhaustive dans l'ensemble de tous les arbres : **computationnellement impossible**
Approche récursive pour construire l'arbre :

construire-arbre(X)

- Si tous les points de X sont de même classe, créer une feuille associée à cette classe
- Sinon
  - choisir (selon critère !) le meilleur couple (attribut, test) pour créer un nœud
  - ce test sépare X en 2 parties Xg et Xd
  - contruire-arbre(Xg)
  - construire-arbre(Xd)

Critère de choix attribut et test

Mesure de l'hétérogénéité du nœud candidat :

soit c_k une classe k ($k = 1, \ldots, K$) et $p(c_k)$ la probabilité de la classe.

Entropie

$$H = -\sum_{k}^{K} p(c_k) log(p(c_k))$$

H=0 si K=1

H=max si $|c_k| = cte \ \forall k$

Indice Gini

$Gini = 1 - \sum_{k}^{K} p(c_k)^2$

Indice d'erreur

$er = 1 - max_k p(c_k)$

Gain d'homogénéité apporté par un test

- Soit un test T à m alternatives et divisant le nœud N en m sous-nœud N_j
- Soit $I(N_j)$ les mesures d'hétérogénéité (entropie, Gini, ...) des sous-nœuds
- Soit $p(N_j)$ les proportions des éléments de N dirigés vers N_j par le test T

  $\Rightarrow$ le gain d'homogénéité apporté par le test T :

$$Gain(N, T) = I(N) - \sum_j p(N_j) I(N_j)$$

  $\Rightarrow$ À chaque nœud, choix du test maximisant le gain

Critères d'arrêt et élagage

- Règles « évidentes » :
  - tous les exemples du nœud sont de même classe
  - tous exemples du nœud ont mêmes valeurs de variables
  - l'hétérogénéité des nœuds ne diminue plus
  - nb d'exemples dans le nœud $<$ seuil minimal
- Contrôle des performances de généralisation (sur base de validation indépendante)
- Elagage a posteriori : supprimer des branches peu représentatives et nuisant à généralisation (parcours bottomup en « remontant » d'un niveau tant que cela diminue erreur en généralisation)

La classification supervisée : "k-plus proches voisins"

Note : une observation x_i est un point dans un espace à p dimensions, appelé espace des descripteurs

Entraînement

exemple : x_n " la valeur des descripteurs audio à D dimensions

on remplit l'espace des descripteurs par l'ensemble des N "points" d'apprentissage : $\{x_1, \ldots, x_N\}$

à chaque point xn est associé sa classe $y_n \in \{c_1, \ldots, c_K\}$

Evaluation :

Soit x_{N+1} une observation nouvelle de classe k inconnue

On recherche dans l'espace des descripteurs les q points les plus proches de x_{n+1} selon une distance , généralement on utilise une distance euclidienne

On associe à x_{n+1} la classe majoritaire parmi celles assignées au K plus proche voisins

$$\hat{x} = m$$

où

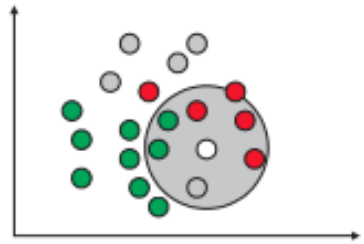
$$m = \underset{k}{argmin} \parallel x - c_k \parallel$$

La classification supervisée : "k-plus proches voisins"

Note : une observation x_i est un point dans un espace à p dimensions, appelé espace des descripteurs

Paramètres :
Choisir le nombre q de plus proches voisins considérés
Choisir le type de distance utilisé (euclidienne ou autres)
Avantage :
Il n'y a pas de modèle à apprendre
Désavantage :
Demande le stockage et l'accès à toutes les données (le nombre de données peut être très très grand)

il faut calculer la distance entre x_{n+1} et tous les point x_i



Bagging : Boostrap Aggregating

Bagging : principe

Principe général : agréger une collection de classifieurs **faibles** pour obtenir un **meilleur** classifieur.

- En général pour la classification, agrégation par vote majoritaire :

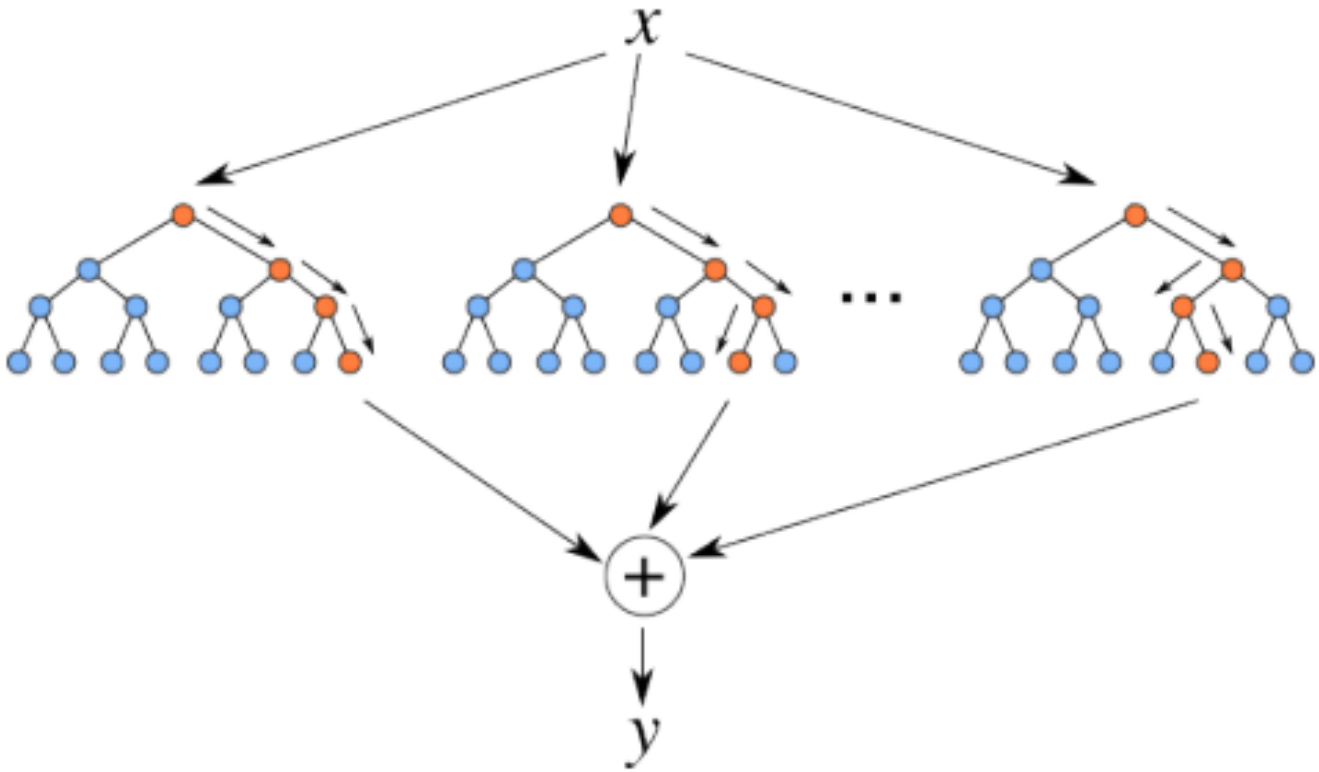
$$\hat{g}_{bag}(x) = \underset{k \in 1,\dots K}{arg\ max} \sum_{l=1}^{B} 1_{\hat{g}_l(x)=k}$$

où $(\hat{g}_l)_{l=1,\dots,B}$ est une collection de B classifieurs

  - $\hat{g}_{bag}(x)$ est correct dès que la majorité des classifieurs $\hat{g}_l$ classe correctement x.
  - Un certain nombre de classifieurs peuvent se tromper sans affecter les performances du classifieur agrégé.

Bagging : principe

Principe général : agréger une collection de classifieurs **faibles** pour obtenir un **meilleur** classifieur.



Bagging : Cas idéal

Dans un contexte de régression $X \in \mathbb{R}^p$ *et* $Y \in \mathbb{R}$ On dispose de B échantillons d'apprentissage iid de taille n : $D_n^{(1)}, \ldots, D_n^{(B)}$ ou $D_n^l = (X_i, Y_i)_{i=1,\ldots,n}^{(l)}, \forall l = 1, \ldots, B$ un échantillon iid.

- On construit les B prédicteurs $\hat{g}_1, \ldots, \hat{g}_B$ iid respectifs à partir des B échantillons d'apprentissage
- Prédicteur agrégé pour la régression : moyenne des prédicteurs

$$\hat{g}_{bag}(x) = \frac{1}{B} \sum_{l=1}^{B} \hat{g}_l(x)$$

Bagging : Cas idéal

Soit g la fonction de régression idéale inconnue :

$$g(x) = \mathbb{E}[Y|X = x]$$

L'erreur commise par le premier prédicteur $\hat{g}_1$ en un point x est :

$$\begin{aligned} E(\hat{g}_1(x) - g(x)^2) = & \quad Var(\hat{g}_1(x)) + (\mathbb{E}(\hat{g}_1(x)) - g(x))^2 \\ = & \quad Var(\hat{g}_1(x)) + Biais^2(\hat{g}_1(x)) \end{aligned}$$

Bagging : Cas idéal

- L'erreur commise par le prédicteur agrégé $\hat{g}_{bag}$ en un point x est :

$$\begin{aligned} E(\hat{g}_{bag}(x) - g(x))^2 = & \quad Var(\hat{g}_{bag}(x) + (E(\hat{g}_{bag}(x)) - g(x))^2 \\ = & \quad Var(\hat{g}_{bag}(x) + Biais^2(\hat{g}_{bag}(x)) \end{aligned}$$

- or $E[\hat{g}_{bag}(x))] = \frac{1}{B}\sum_{l=1}^{B} E[\hat{g}_l(x)] = E[\hat{g}_1(x))]$ car les $\hat{g}_l$ sont iid
  - $$\text{et } Var(\hat{g}_{bag}(x) = \frac{1}{B^2}\sum_{l=1}^{B} Var[\hat{g}_l(x)] = \frac{1}{B^2}\sum_{l=1}^{B} Var[\hat{g}_1(x)]$$
  $$E(\hat{g}_{bag}(x) - g(x))^2 = \frac{1}{B} Var[\hat{g}_1(x)] + Biais^2(\hat{g}_1(x))$$
  - Donc si B est grand, on diminue fortement la variance et donc l'erreur du prédicteur agrégé est beaucoup plus faible.

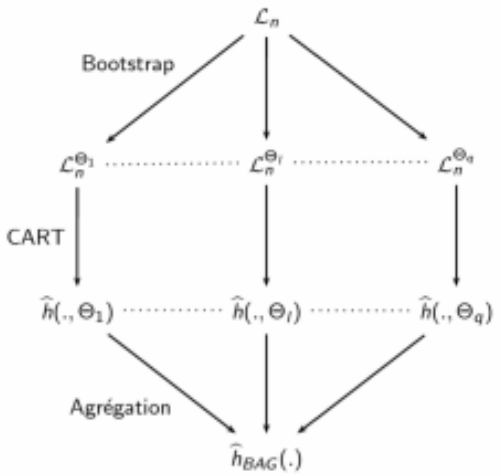
Bagging : Cas pratique

- En réalité, on ne dispose pas de B échantillons d'apprentissage iid de taille n, avec B grand.
- L'approche bagging consiste à tenter d'atténuer la dépendance entre les estimateurs que l'on agrège
- Solution : générer B échantillons à partir de l'échantillon de départ.

Bagging : Méthode

1. Bootstrap : on construit B échantillons $D_n^1, \ldots, D_n^B$ à partir d'un seul échantillon de départ $D_n = (X_i, Y_i)_{i=1,\ldots,n}$.
   - Un échantillon bootstrap $D_n^{(l)}$ est obtenu par tirage avec remise de n éléments parmi D_n où chaque observation (X_i, Y_i) a une probabilité $\frac{1}{n}$ d'être tirée à chaque tirage.
2. Chaque échantillon bootstrap $D_n^{(l)}$ sert à construire un classifieur $\hat{g}_l \ \forall = 1, \ldots, B$
3. On agrège cette collection de classifieurs $\hat{g}_1, \ldots, \hat{g}_B$:

$$\hat{g}_{bag}(x) = \underset{k \in 1,\ldots K}{arg\ max} \sum_{l=1}^{B} 1_{\{\hat{g}_l(x)=k\}}$$



Bagging : L'algorithme

1. Entrées :
   - x l'observation à prévoir
   - un régresseur (arbre CART, 1 plus proche voisin...)
   - $\mathcal{D}_n$ l'échantillon
   - B le nombre d'estimateurs que l'on agrège.
2. Pour $l = 1, \ldots, B$
3. Tirer un échantillon bootstrap $\mathcal{D}_n^l$ dans $\mathcal{D}_n$
4. Ajuster le régresseur sur cet échantillon bootstrap : $\hat{g}_l$
5. Sortie l'estimateur

$$\hat{g}_{bag}(x) = \frac{1}{B}\sum_{k=1}^{B}\hat{g}_l(x)$$

La loi forte des grands nombre :

$$\lim_{B\longrightarrow\infty}\hat{g}_{bag}(x) = \lim_{B\longrightarrow\infty}\frac{1}{B}\sum_{l=1}^{B}\hat{g}_{bag}(x) = E_\theta(\hat{g}_{bag}(x)|\mathcal{D}_n) \text{ P.S}$$

Bagging : remarques

- Les classifieurs de la collection $\hat{g}_1, \ldots, \hat{g}_B$ ne sont pas indépendants car les échantillons ne le sont pas. Cependant le Bagging agit comme un réducteur de variance.
- Contrairement au boosting, prendre B trop grand ne va pas sur-ajuster l'échantillon.
- Le choix de B n'est donc pas crucial pour la performance de l'estimateur, il est recommandé de le prendre le plus grand possible (en fonction du temps de calcul).
- Bagger des pédicteurs stables n'apporte rien, il faut des classifieurs suffisamment différents les uns des autres pour améliorer les performances par Bagging

Biais et variance

- L'estimateur agrégé

$$\hat{g}_B(x) = \frac{1}{B}\sum_{l=1}^{B}\hat{g}_l(x) \text{ et } \hat{g}(x) = \underset{B\longrightarrow\infty}{\hat{g}_B(x)}$$

- $\sigma^2(x) = V(\hat{m}(x))$ La variance des estimateurs que l'on agrège
- $\rho(x) = corr[\hat{g}_l(x), \hat{g}_m(x)]$ le coefficient de corrélation entre deux estimateurs que l'on agrège (calculés sur deux échantillons bootstrap).

Biais

On suppose que les estimateurs $\hat{g}_1(x), \ldots, \hat{g}_B(x)$ sont identiquement distribués. le biais de l'estimateur agrégé est le même que le biais des estimateurs que l'on agrège.
Par conséquent agréger ne modifie pas le biais.

Biais et variance

- L'estimateur agrégé

$$\hat{g}_B(x) = \frac{1}{B}\sum_{l=1}^{B}\hat{g}_l(x) \text{ et } \hat{g}(x) = \underset{B\longrightarrow\infty}{\hat{g}_B(x)}$$

- $\sigma^2(x) = V(\hat{g}_l(x))$ La variance des estimateurs que l'on agrège
- $\rho(x) = corr[\hat{g}_1(x), \hat{g}_2(x)]$ le coefficient de corrélation entre deux estimateurs que l'on agrège (calculés sur deux échantillons bootstrap).

Variance

On a :

$$V(\hat{g}_B(x)) = \rho(x)\sigma^2(x) + \frac{1-\rho(x)}{B}\sigma^2(x)$$

Par conséquent

$$V(\hat{g}(x)) = \rho(x)\sigma^2(x)$$

. Ainsi : si $\rho(x) < 1$, l'estimateur baggé a une variance plus petite que celle des estimateurs que l'on agrège (pour B suffisamment grand).

Biais et variance

- C'est la corrélation $\rho(x)$ entre les estimateurs que l'on agrège qui quantifie le gain de la procédure d'agrégation
- Il faut que les estimateurs que l'on agrège soient sensibles à des perturbations de l'échantillon par bootstrap
- Les arbres de **régression** et de **classification** sont des estimateurs connus pour être instables.

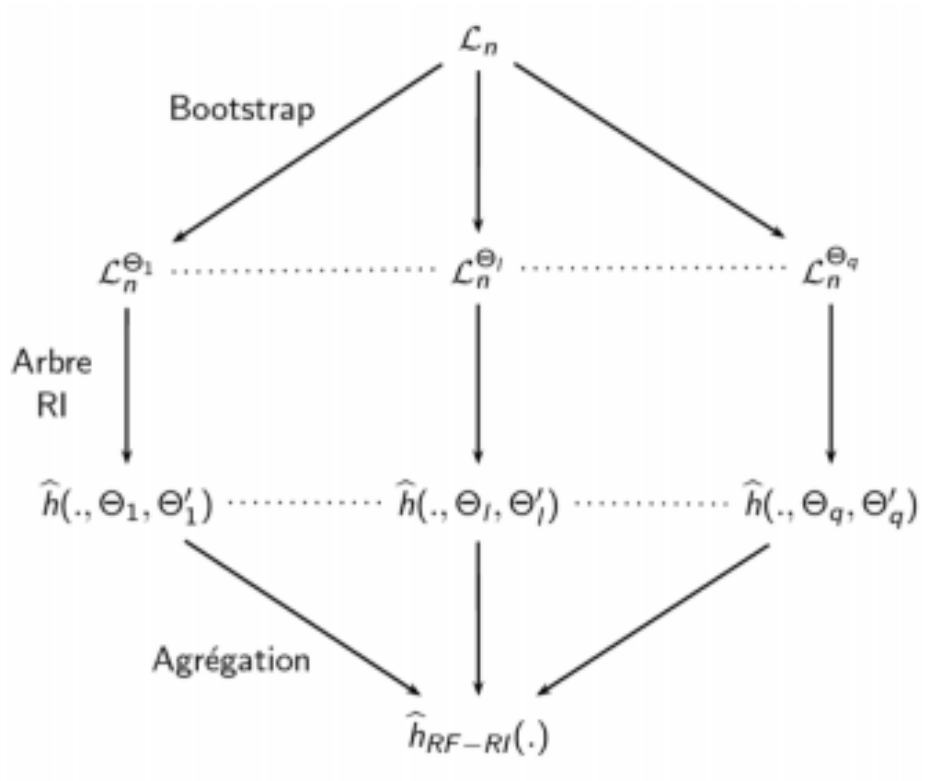
Forêt Aléatoire

Forêts aléatoires : principe

- Méthode d'ensemble appliquée et limitée aux arbres CART.
- Principe : décorréler la collection d'arbres provenant du Bagging en introduisant de l'aléa dans la construction de l'arbre.

  plusieurs sources d'aléa ont été testées mais la méthode de Breiman (RF-RI) s'est imposée comme la méthode RF (Random Forest) par excellence

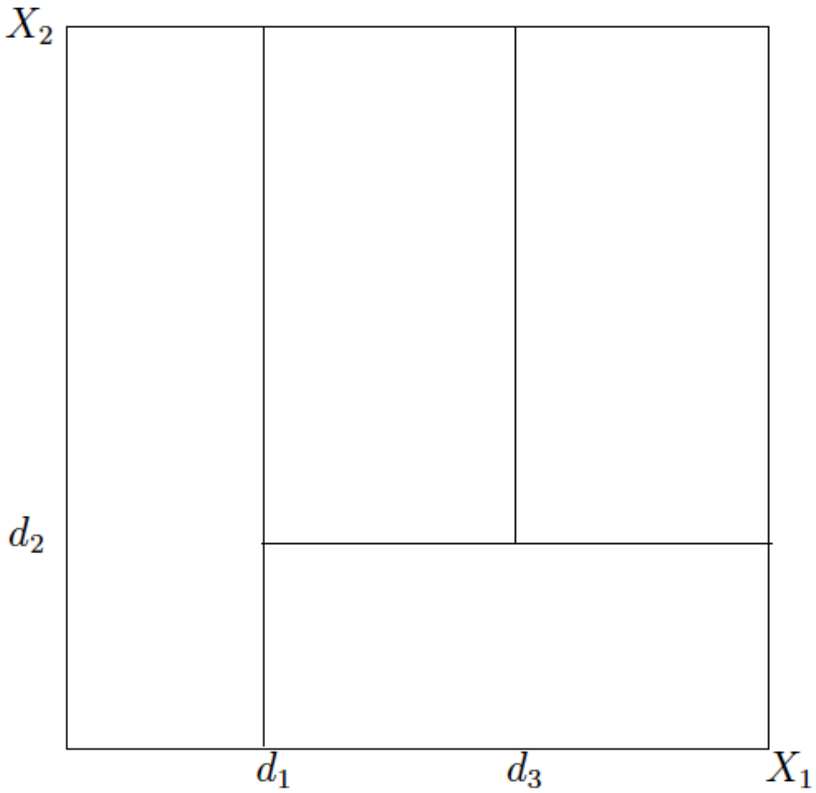
Forêts aléatoires : méthode

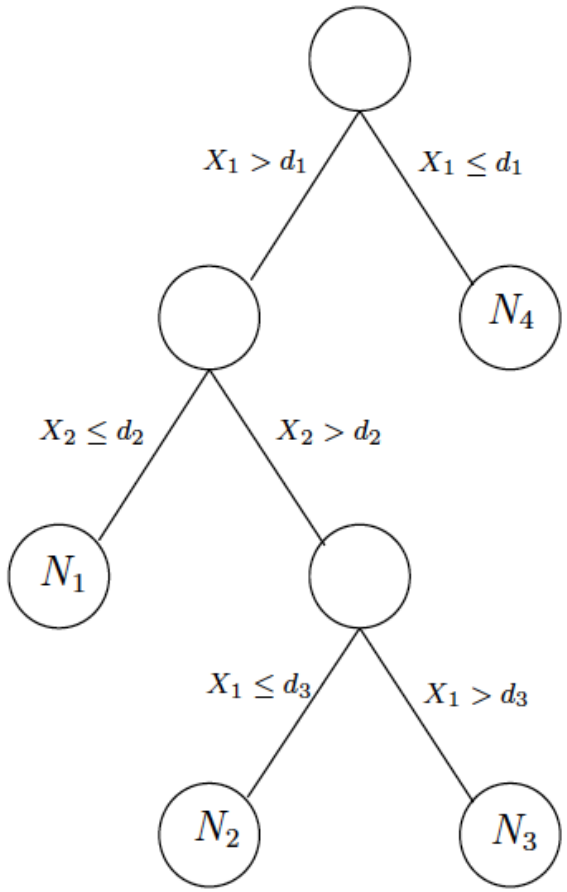


Notations :

- $\theta_l, l = 1, \ldots, B$ sont des variables indépendantes pour l'aléa de bootstrap.
- θ_l', $l = 1, ..., B$ sont des variables indépendantes pour l'aléa du prédicteur
- $\hat{h}(., \theta_l, \theta_0)$ est le prédicteur aléatoire construit à partir de l'échantillon $\mathcal{D}_n^{\theta_l}$

CART





A chaque étape, on cherche la variable X^j et le réel d qui minimisent :
— la variance des noeuds fils en régression ;
— l'indice de Gini des noeuds fils en classification.

Forêts aléatoires : méthode de Breiman

1. On tire B échantillons bootstrap $\mathcal{D}_n^{(1)}, \ldots, \mathcal{D}_n^{(B)}$ (comme pour le Bagging)
2. Sur chaque $\mathcal{D}_n^{(l)}$, on applique une variante de CART appelée RI (Random Input) qui diffère à 2 niveaux :
    1. Pour découper un noeud de l'arbre, on optimise la coupure sur un ensemble aléatoire de $m \leq p$ variables (tirage sans remise de m parmi p variables)
    2. Les arbres ne sont pas élagués, on garde l'arbre maximal à chaque fois

    Remarque : m est fixé pour tous les noeuds de l'arbre
3. Enfin, on agrège la collection de classifieurs obtenus $\hat{g}_1, \ldots, \hat{g}_B$

$$\hat{g}_{RF} = \underset{k \in \{1,\ldots,K\}}{argmax} \sum_{l=1}^{B} 1_{\{\hat{g}_l(x)=k\}}$$

Remarques

- Dans un contexte de discrimination, l'étape finale d'agrégation dans l'algorithme consiste à faire voter les arbres à la majorité

- Compromis biais-variance dans le choix de m

  - Lorsque m diminue, la corrélation entre les arbres va avoir tendance à diminuer également. Conséquence : une baisse de la variance de l'estimateur agrégé

  - Choisir les axes de découpe des arbres de manière (presque) aléatoire va se traduire par une moins bonne qualité d'ajustement des arbres sur l'échantillon d'apprentissage, d'où une augmentation du biais pour chaque arbre ainsi que pour l'estimateur agrégé.

  - lorsque m augmente, les phénomènes inverses se produisent.

Concernant le choix de m, randomForest propose par défaut $m = p/3$ en régression et $m = \sqrt{p}$ en classification

Erreur Out Of Bag : OOB

Taux d'erreur Out Of Bag : estimation acceptable du taux d'erreur théorique obtenue grâce au bootstrap (sans CV)

- Pour $i = 1, ..., n$: on considère l'observation $(X_i, Y_i) \in \mathcal{D}_n$
  - on considère l'ensemble des échantillons bootstrap ne contenant pas (X_i, Y_i) (pour lesquels (X_i, Y_i) est OOB = "en dehors du bootstrap") ainsi que tous les arbres associés à ces échantillons.
  - on prédit $\hat{Y}_i$ en fonction de X_i en agrégeant uniquement ces arbres
- Erreur Out Of Bag

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \text{ en Regression}$$

$$\frac{1}{n}\sum_{i=1}^{n}1_{(\hat{Y}_i \neq Y_i)} \text{ en Classification}$$

L'avantage de la procédure Out Of Bag (OOB) est qu'elle ne nécessite pas de découper l'échantillon.

Procédure Out Of Bag

Taux d'erreur Out Of Bag : estimation acceptable du taux d'erreur théorique obtenue grâce au bootstrap (sans CV)
Soit une observation $(X_i, Y_i) \in \mathcal{D}_n$

On désigne par $\mathcal{I}_B$ l'ensemble des arbres de la forêt ne contenant pas i dans leur échantillon bootstrap

On détermine $\hat{Y}_i$:

$$\hat{Y}_i = \frac{1}{|\mathcal{I}_B|} \sum_{k \in \mathcal{I}_B} h(X_i, \theta_k)$$

Erreur Out Of Bag

$$\frac{1}{n} \sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2 \text{ en Regression}$$

$$\frac{1}{n} \sum_{i=1}^{n} 1_{(\hat{Y}_i \neq Y_i)} \text{ en Classification}$$

L'avantage de la procédure Out Of Bag (OOB) est qu'elle ne nécessite pas de découper l'échantillon.

Importance des variables

On désigne par OOB_k l'échantillon Out Of Bag associé au k^{eme} arbre de la forêt. Cet échantillon est formé par les observations qui ne figurent pas dans le k^{eme} échantillon bootstrap

- E_{OOB_k} l'erreur de prédiction de l'arbre $h(., \theta_k)$

$$E_{OOB_k} = \frac{1}{|OOB_k|} \sum_{i \in OOB_k} (h(X_i, \theta_k) - Y_i)^2$$

- $E_{OOB_k^j}$ l'erreur de prédiction de l'arbre $h(., \theta_k)$ dans lequel on a perturbé aléatoirement les valeurs de la variable j

$$E_{OOB_k^j} = \frac{1}{|OOB_k^j|} \sum_{i \in OOB_k^j} (h(X_i^j, \theta_k) - Y_i)^2$$

-

$$Imp(X^j) = \frac{1}{B} \sum_{k}^{B} (E_{OOB_k^j} - E_{OOB_k})$$

Importance des variables

Heuristiquement

- si la jème variable joue un rôle déterminant dans la construction de l'arbre $h(.,\theta_k)$, alors une permutation de ces valeurs dégradera fortement l'erreur
- La différence d'erreur

$$E^j_{OOB_k} - E_{OOB_k}$$

  sera alors élevée

Exemple par la pratique

Données : 4 601 observations et 57 variables explicatives

Y qui prend pour valeur 1 si le mail est un spam, 0 sinon.

$ftp : //ftp.ics.uci.edu/pub/machine - learning - databases/spambase/$

1. Charger les données et diviser l'échantillon en test (50%) et apprentissage (50%)
2. Appliquer : $randomForest(Y \sim ., data = dapp1)$ et déterminer les variables importantes
3. Calculer l'erreur OOB
4. Comparer cette erreur avec l'erreur estimée sur l'échantillon de validation (test)

Boosting

Contexte de classification supervisée

- Y prend ses valeurs dans $\{-1, 1\}$
- Soit g la règle d'affection de **performance** $L(g) = P(g(X) \neq Y)$
- Règle de Bayes

$$g^*(x) = \begin{cases} 1 & \text{si } P(Y = 1|X = x) > 0.5, \\ -1 & \text{sinon} \end{cases}$$

- $$L^* = L(g^*(x)) = \inf_g L(g)$$

Il est bien évidemment impossible de calculer $g*$ en pratique.

Solution : Construire un estimateur $g_n(x) = g_n(X, D_n)$ tel que $L(g_n)$ qui soit le proche possible de L^*

En ce sens, on dira que la suite (g_n) est convergente pour une certaine distribution (X, Y) si $\lim_{n\to\infty} \mathrm{E}L\,(g_n) = L^\star$.

Si la suite (g_n) est consistante pour toutes les distributions de (X, Y), on parlera de consistance universelle.

Boosting

Exemple : K-PPV : Pour $k \leq n$

$$g_n^{ppv}(x) = \begin{cases} 1 & \text{si } \sum_{i=1}^{k} 1_{Y_{(i)}=1} > \sum_{i=1}^{k} 1_{Y_{(i)}=-1} \\ -1 & \text{sinon} \end{cases}$$

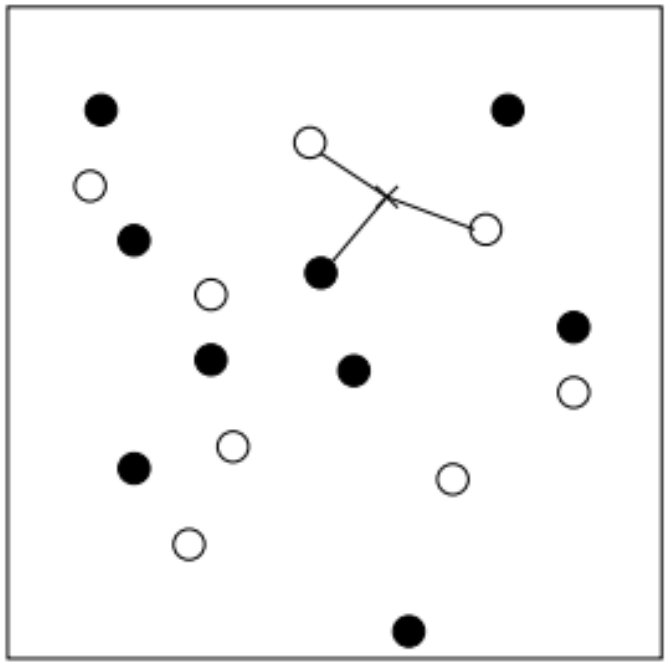


Figure – Source : Rouvière

Si $k = k_n \longrightarrow \infty$ et $\frac{k}{n} \longrightarrow 0$ quand $n \longrightarrow \infty$ alors la règle des plus proches voisins est **universellement convergentes**

Boosting

Difficultés : à distance finie

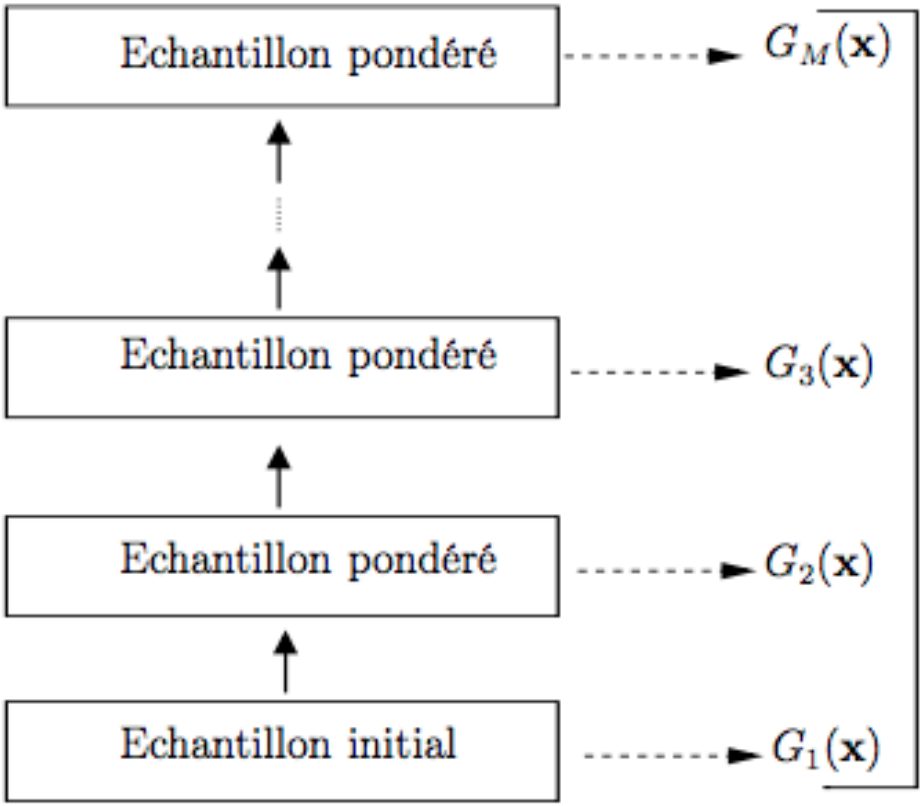
- Le choix des paramètres : nombre de plus proches voisins impacte les performances de ces estimateurs.
- Fléau de la dimension

Solution : Les méthodes d'agrégation permettent de pallier à certaines de ces difficultés.

Boosting

- Le principe général du boosting consiste à construire de manière récursive une famille d'estimateurs
- chaque estimateur est une version adaptative du précédent en donnant plus de poids aux observations mal ajustées ou mal prédites.
- L'estimateur construit à l'étape k concentrera donc ses efforts sur les observations mal ajustées par l'estimateur à l'étape $k-1$.
- qui sont ensuite agrégés par une moyenne pondérée des estimations (en régression) ou un vote à la majorité (en discrimination).

L'algorithme adaboost



$$G(\mathbf{x}) = \text{signe}\left[\sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})\right]$$

Représentation de l'algorithme adaboost

L'algorithme adaboost

$G(x)$ une règle de classification "faible" (weaklearner).

Algorithme 1 AdaBoost

Entrée :

- $\mathbf{x}$ l'observation à prévoir
- $d_n = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ l'échantillon
- Une réglè faible
- M le nombre d'itérations.

1. Initialiser les poids $w_i = 1/n$, $i = 1, \ldots, n$
2. **Pour** $m = 1$ à M :
   (a) Ajuster la règle faible sur l'échantillon d_n pondéré par les poids $w_1, \ldots, w_n$, on note $g_m(\mathbf{x})$ l'estimateur issu de cet ajustement
   (b) Calculer le taux d'erreur :
   $$e_m = \frac{\sum_{i=1}^{n} w_i 1_{y_i \neq g_m(\mathbf{x}_i)}}{\sum_{i=1}^{n} w_i}.$$
   (c) Calculer : $\alpha_m = \log((1 - e_m)/e_m)$
   (d) Réajuster les poids :
   $$w_i = w_i \exp(\alpha_m 1_{y_i \neq g_m(\mathbf{x}_i)}), \quad i = 1, \ldots, n$$
3. **Sortie** : $\hat{g}_M(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m g_m(\mathbf{x})$.

L'algorithme adaboost

Dans le contexte de la classification binaire on cherche à minimiser l'espérance d'une fonction de perte $\ell : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$:

$$g^*(x) = \underset{g \in \mathcal{G}}{argmin}\ E[\ell(Y, g(X))]$$

En classification on peut poser naturellement $\ell(Y, g(X)) = 1_{\{g(X) \neq y\}}$

La loi de X et Y étant inconnue on minimise la version empirique

$$g^*(x) = \underset{g \in \mathcal{G}}{argmin} \frac{1}{n} \sum_i 1_{(y_i \neq g(x_i))}$$

L'estimateur adaboost

Soit $\mathcal{G}$ une classe de règles faibles. On cherche $g \in \mathcal{G}$ qui minimise l'erreur empirique. Alors l'estimateur adaboost à l'étape M s'écrit

$$\hat{g}(M) = \hat{g}(M-1) + 2\beta_M G_M$$

avec

$$(\beta_M, G_M) = \underset{(\beta,g)\in\mathbb{R}\times G}{argmin} \sum_{i=1}^{n} exp(-y_i(\hat{g}_{M-1}(x_i) + \beta g(x_i))).$$

$$\beta_M = \frac{1}{2} log(\frac{1 - err_M}{err_M}) \text{ et } err_M = \frac{\sum_{i=1}^{n} w_i^M 1_{y_i \neq G_M(x_i)}}{\sum_{i=1}^{n} w_i^M}$$

$$w_i^{M+1} = w_i^M exp(2\beta_M 1_{y_i \neq G_M(x_i)}) exp(-\beta_M).$$

C.f : (Hastie et al (2009) (chapitre 10))

Généralisation

Algorithme 2 Boosting par descente de gradient fonctionnelle.

Entrées :

- $\mathbf{x}$ l'observation à prévoir
- h une règle faible
- $d_n = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ l'échantillon
- λ un paramètre de régularisation tel que $0 < \lambda \leq 1$.
- M le nombre d'itérations.

1. Initialisation :

$$g_0(.) = \underset{c}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, c)$$

2. **Pour** $m = 1$ à M :

   (a) Calculer l'opposé du gradient $-\frac{\partial}{\partial g} L(y, g)$ et l'évaluer aux points $g_{m-1}(\mathbf{x}_i)$:

$$U_i = - \left[\frac{\partial \ell(y_i, g(\mathbf{x}_i))}{\partial g(\mathbf{x}_i)} \right]_{\Big| g(\mathbf{x}_i) = g_{m-1}(\mathbf{x}_i)}, \quad i = 1, \ldots, n.$$

   (b) Ajuster la règle faible sur l'échantillon $(\mathbf{x}_1, U_1), \ldots, (\mathbf{x}_n, U_n)$, on note h_m la règle ainsi définie.

   (c) Mise à jour : $g_m(\mathbf{x}) = g_{m-1}(\mathbf{x}) + \lambda h_m(\mathbf{x})$.

3. **Sortie :** la règle $g_M(\mathbf{x})$.

Exemple

Simuler un échantillon d'apprentissage de taille 100 selon le modèle :

- $X_i \sim N(0,1)$, $U_i \sim U[0,1]$ et

$$Y_i = \begin{cases} 1_{U_i \leq 0.25} & \text{si } X_i \leq 0, \\ 1_{U_i > 0.25} & \text{si } X_i > 0, \end{cases}$$

- Calculer l'erreur de Bayes

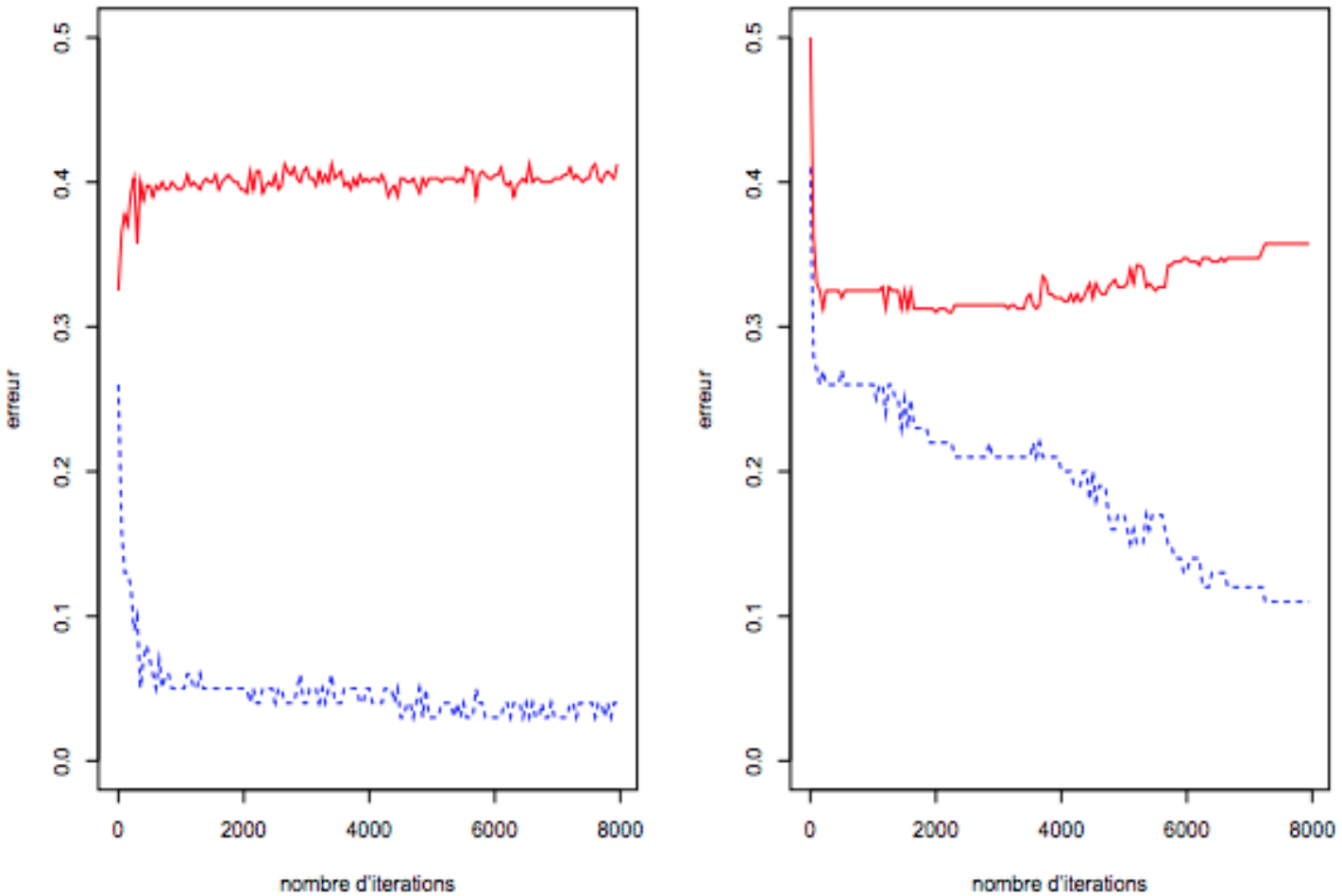
- Pour $\lambda = 1$ et 0.01 lancer
  $model < -gbm(Y\ ., data = dapp, distribution = "adaboost", interaction.depth = 1, shrinkage = 1, n.trees = B)$

- Estimer le pourcentage de mal classé sur les échantillons test et d'apprentissage.

Exemple

```
> boucle <- seq(1,B,by=50)
> errapp <- rep(0,length(boucle))
> errtest <- errapp
> k <- 0
> for (i in boucle){
+  k <- k+1
+  prev_app <- predict(model,newdata=dapp,n.trees=i)
+  errapp[k] <- sum(as.numeric(prev_app>0)!=dapp$Y)/nrow(dapp)
+  prev_test<- predict(model,newdata=dtest,n.trees=i)
+  errtest[k] <- sum(as.numeric(prev_test>0)!=dtest$Y)/nrow(dtest)
+}
```

Exemple



Taux de mal classés estimés sur l'échantillon test (rouge, trait plein) et sur l'échantillon d'apprentissage (bleu, tirets) pour $\lambda = 1$ (gauche) et $\lambda = 0.01$ (droite).

Logitboost

Considérons le modèle logistique :

$$p(x) = \frac{1}{1 + exp(-X'\beta)} = \frac{exp(X'\beta)}{1 + exp(X'\beta)}$$

Le vecteur de paramètres β est généralement estimé en maximisant la vraisemblance.
Dans le cas du modèle **Logiboost**

$$p(x) = \frac{1}{1 + exp(-2f(x))} \text{ avec } f(x) = \frac{1}{2} log(\frac{p(x)}{1 - p(x)})$$

la fonction f qui maximise la log-vraisemblance

$$-(ylog(p(x)) + (1 - y)log(1 - p(x))) = log(1 + exp(-2\tilde{y}f)) \text{ Ainsi } \tilde{y} = 2y - 1$$

L'approche logitboost consiste à appliquer l'algorithme généralisation en utilisant la fonction de perte

$$L(y, f) = log(1 + exp(-2\tilde{y}f))$$

Logitboost

Après M itérations, l'algorithme fournit un estimateur f_M de

$$f^*(.) = \underset{f}{argmin}\ E[log(1 + exp(-2Y\tilde{f}(X)))].$$

- On montre que
$$f^*(x) = \frac{1}{2} log(\frac{p(x)}{1 - p(x)})$$
- On en deduit
$$p_M(x) = \frac{1}{1 + exp(-2f_M(X'))}$$
- Règle de classification
$$\hat{y} = \begin{cases} 1 & \text{si } f_M(X) \leq 0, \\ 0 & \text{si } f_M(X) > 0 \end{cases}$$

L_2 boosting

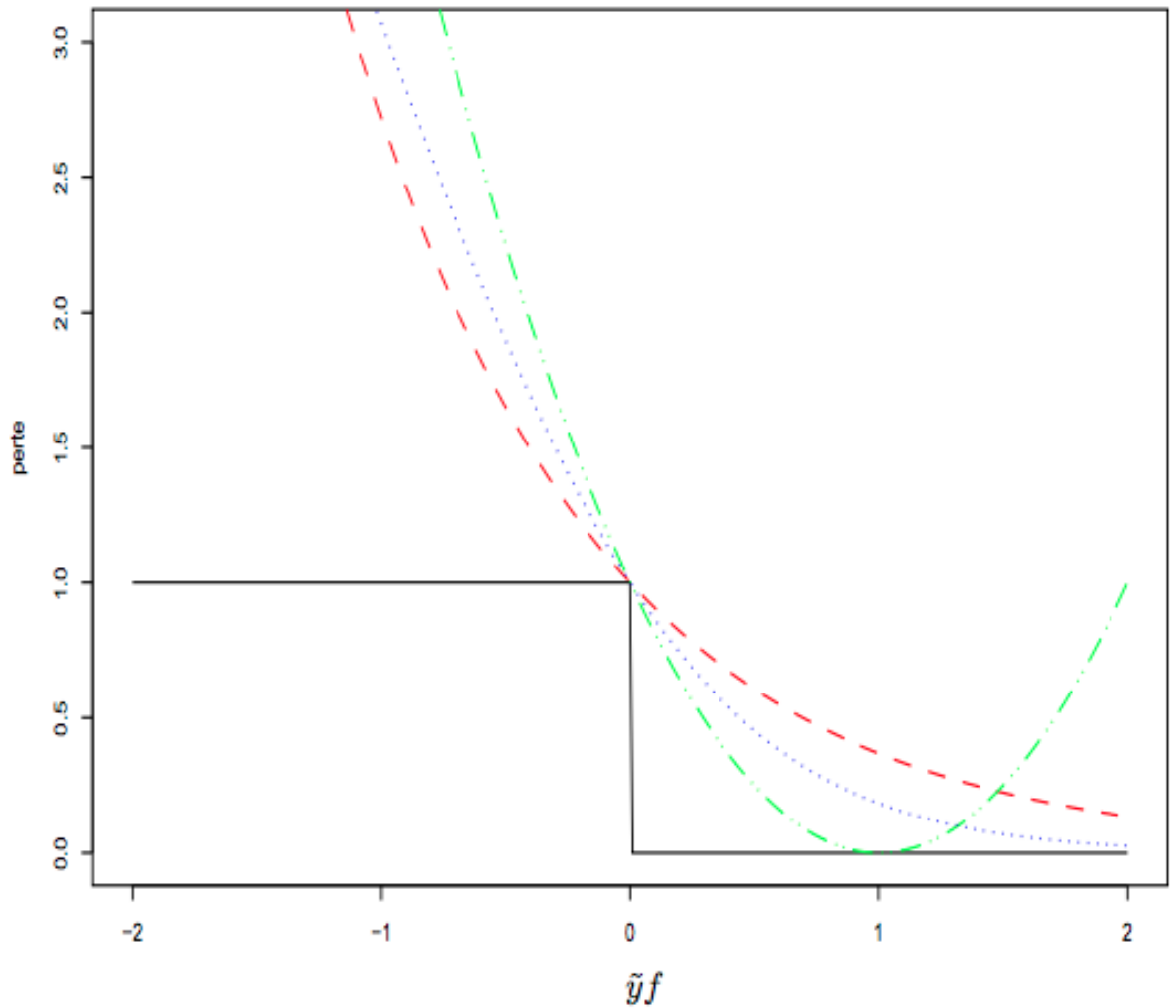
L_2 boosting consiste à appliquer l'algorithme Généralisation avec la fonction de perte quadratique

$$L(y, g) = (y - g)^2$$

Algorithmes boosting : $\tilde{y} = 2y - 1$

Espaces	$\ell(y, f)$	$f^*(\mathbf{x})$	Algorithme
$y \in \{0, 1\},\ f \in \mathbb{R}$	$\exp(-2\tilde{y}f)$	$\frac{1}{2}\log\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right)$	Adaboost
$y \in \{0, 1\},\ f \in \mathbb{R}$	$\log(1 + \exp(-2\tilde{y}f))$	$\frac{1}{2}\log\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right)$	Logitboost
$y \in \mathbb{R},\ f \in \mathbb{R}$	$\frac{1}{2}(y - f)^2$	$\mathbf{E}[Y \mid \mathbf{X} = \mathbf{x}]$	L_2 boosting

Exemple



Fonctions de perte des algorithmes boosting : adaboost (rouge, tirets), logitboost (bleu, pointillés), L_2 boosting (vert, points-tirets). La fonction de perte $1_{yf>0}$ est en noir (trait plein).