

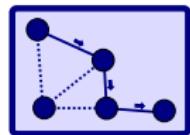
Une approche d'optimisation discrète pour la classification associative

SOD322 - RO et données massives

Zacharie ALES
(zacharie.ales@ensta-paristech.fr)

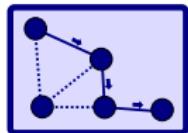
Créé le 10/01/2018
Modifié le 21/01/2020 (v3)





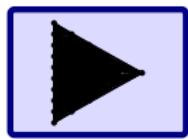
Introduction à la RO

 O. Klopfenstein



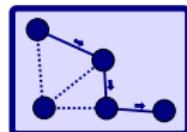
Introduction à la RO

☞ O. Klopfenstein

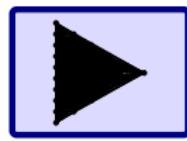


Introduction au ML

☞ P. Bianchi



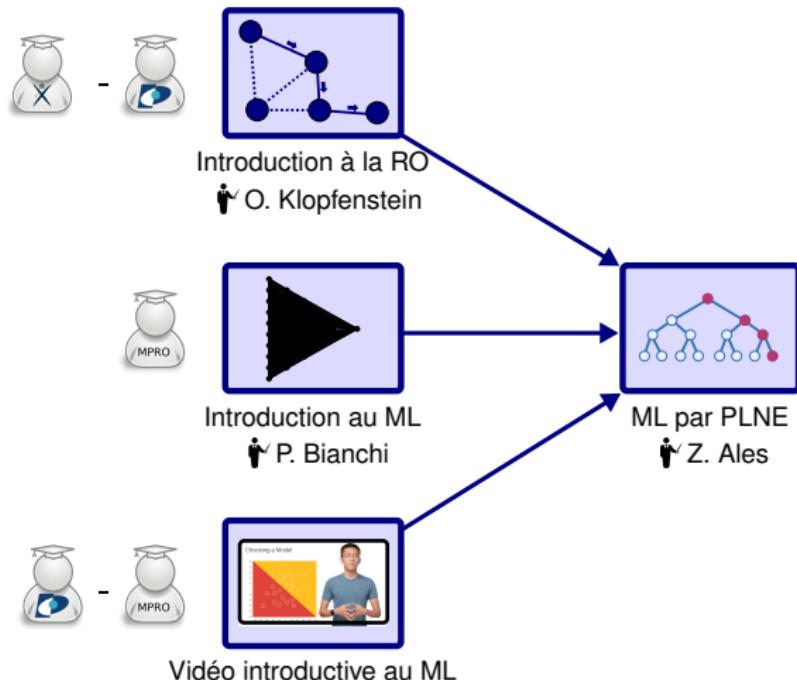
Introduction à la RO
O. Klopfenstein

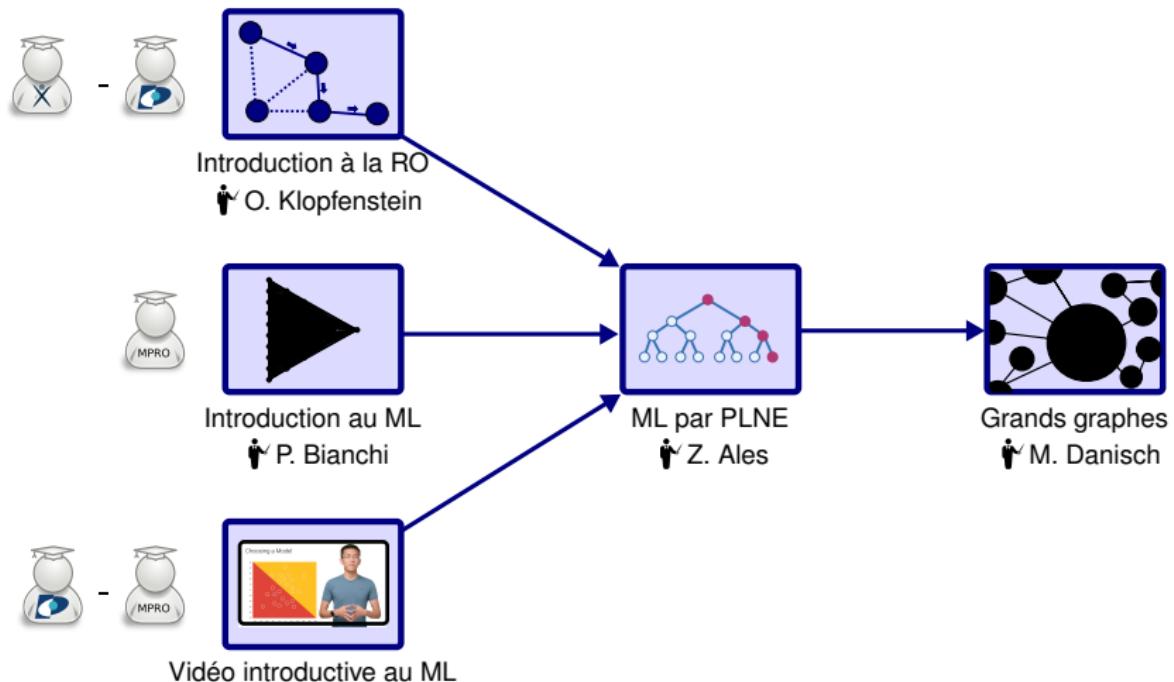


Introduction au ML
P. Bianchi



Vidéo introductive au ML





1 Introduction au machine learning

2 ORC - Règles ordonnées pour la classification

- Exemple introductif
- Étape 1 - Génération de règles
- Étape 2 - Classement des règles générées
- Résultats

3 Projet

- Présentation du sujet
- Julia

Sommaire

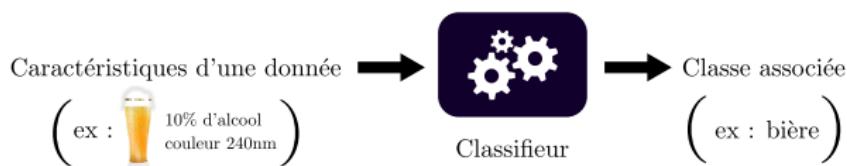
1 Introduction au machine learning

2 ORC - Règles ordonnées pour la classification

3 Projet

Définition - Apprentissage automatique ("machine learning")

Conception et analyse d'algorithmes capable d'apprendre à partir d'exemples



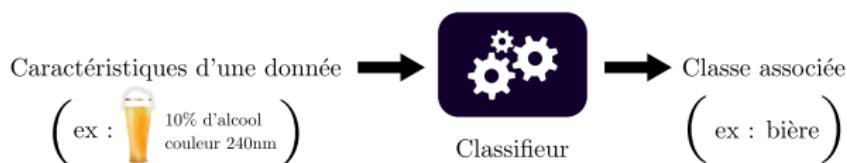
Définition - Apprentissage automatique ("machine learning")

Conception et analyse d'algorithmes capable d'apprendre à partir d'exemples

Définition - Classification supervisée

Conception et analyse d'algorithmes visant à étiqueter individuellement des données

Associer une classe à chaque donnée en fonction de ses caractéristiques



Définition - Apprentissage automatique ("machine learning")

Conception et analyse d'algorithmes capable d'apprendre à partir d'exemples

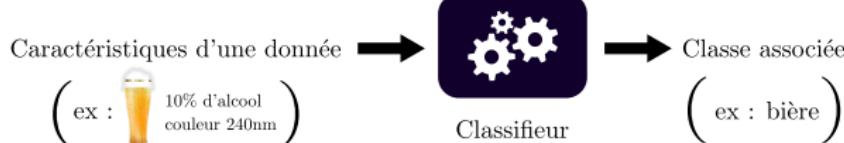
Définition - Classification supervisée

Conception et analyse d'algorithmes visant à étiqueter individuellement des données

Associer une classe à chaque donnée en fonction de ses caractéristiques

Exemple - Classe

- Vin, bière
- Chat, chien, oiseaux, ...
- Reconnaissance de chiffres manuscrits
- ...



Définition - Apprentissage automatique ("machine learning")

Conception et analyse d'algorithmes capable d'apprendre à partir d'exemples

Définition - Classification supervisée

Conception et analyse d'algorithmes visant à étiqueter individuellement des données

Associer une classe à chaque donnée en fonction de ses caractéristiques

Exemple - Classe

- Vin, bière
- Chat, chien, oiseaux, ...
- Reconnaissance de chiffres manuscrits
- ...

Exemple - Caractéristiques

- Taux d'alcool, couleur
- Ratio hauteur/longueur, forme, ...

Caractéristiques d'une donnée

(ex :  10% d'alcool
couleur 240nm)



Classifieur

Classe associée

(ex : bière)

Définition - Apprentissage automatique ("machine learning")

Conception et analyse d'algorithmes capable d'apprendre à partir d'exemples

Définition - Classification supervisée

Conception et analyse d'algorithmes visant à étiqueter individuellement des données

Associer une classe à chaque donnée en fonction de ses caractéristiques

Exemple - Classe

- Vin, bière
- Chat, chien, oiseaux, ...
- Reconnaissance de chiffres manuscrits
- ...

Exemple - Caractéristiques

- Taux d'alcool, couleur
- Ratio hauteur/longueur, forme, ...

Définition - Classifieur

Algorithme de classification

Caractéristiques d'une donnée

(ex :  10% d'alcool
couleur 240nm)



Classifieur

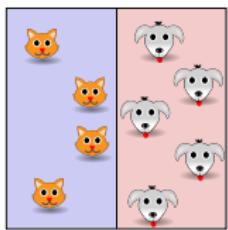


Classe associée

(ex : bière)

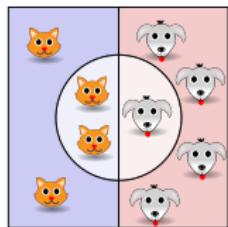
Comment évaluer un classifieur ?

Images de chiens et de chats



Comment évaluer un classifieur ?

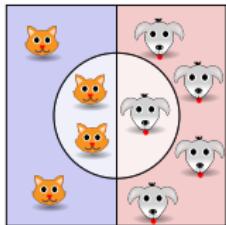
Images de chiens et de chats



: Images de chats selon le classifieur

Comment évaluer un classifieur ?

Images de chiens et de chats



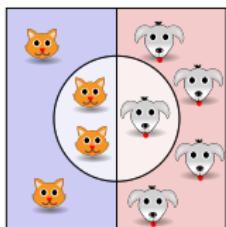
: Images de chats selon le classifieur

Définitions (point de vue de la classe chat)

- : vrais positifs (VP) Chats prédisits chat
- : faux positifs (FP) Chats prédisits chat
- : faux négatifs (FN) Chats prédisits chat
- : vrai négatifs (VN) Chats prédisits chat

Comment évaluer un classifieur ?

Images de chiens et de chats



: Images de chats selon le classifieur

Définitions (point de vue de la classe chat)

- : vrais positifs (VP) Chats prédis chat
- : faux positifs (FP) Chats prédis chat
- : faux négatifs (FN) Chats prédis chat
- : vrai négatifs (VN) Chats prédis chat

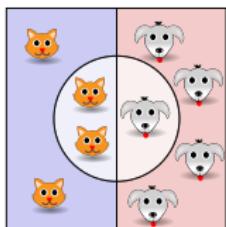
Définition - Précision d'une classe

$$\frac{VP}{VP + FP}$$

Proportion d'animaux prédis chats qui sont bien des chats ($\frac{2}{3}$ dans l'exemple)

Comment évaluer un classifieur ?

Images de chiens et de chats



: Images de chats selon le classifieur

Définitions (point de vue de la classe chat)

- : vrais positifs (VP) Chats prédis chat
- : faux positifs (FP) Chats prédis chat
- : faux négatifs (FN) Chats prédis chat
- : vrai négatifs (VN) Chats prédis chat

Définition - Précision d'une classe

$$\frac{VP}{VP + FP}$$

Proportion d'animaux prédis chats qui sont bien des chats ($\frac{2}{3}$ dans l'exemple)

Définition - Rappel d'une classe

$$\frac{VP}{VP + FN}$$

Proportion de chats bien prédis ($\frac{2}{4}$ dans l'exemple)

Définition - Précision d'un classifieur

Moyenne des précisions de toutes les classes

Eventuellement pondérée par le nombre de prédictions de chaque classe

Définition - Rappel d'un classifieur

Moyenne des rappels de toutes les classes

Eventuellement pondérée par le nombre d'éléments de chaque classe

Partage des données

- ① **Apprentissage** ("train") : données utilisées pour définir le classifieur
- ② **Test** : données utilisées pour évaluer les performances du classifieur

Une évaluation possible d'un classifieur

Précision et rappel sur les données d'apprentissage et de test

Exemple de classifieurs

Classifieur - Arbres de décision

Arbre dont

- les noeuds internes sont des choix
- les feuilles sont des classes

Exemple - Le client a-t-il des chances d'acheter un ordinateur ?

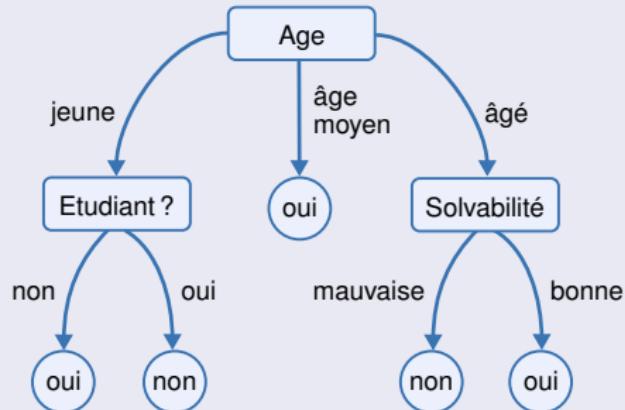


Image inspirée de tutorialspoint.com

Exemple de classifieurs

Classifieur - Forêts d'arbres décisionnels

- Apprentissage de multiples arbres aléatoires sur des sous-ensembles de données légèrement différents
- Prédiction : vote majoritaire des arbres

Aussi appelées forêts aléatoires ("random forest classifier" en anglais)

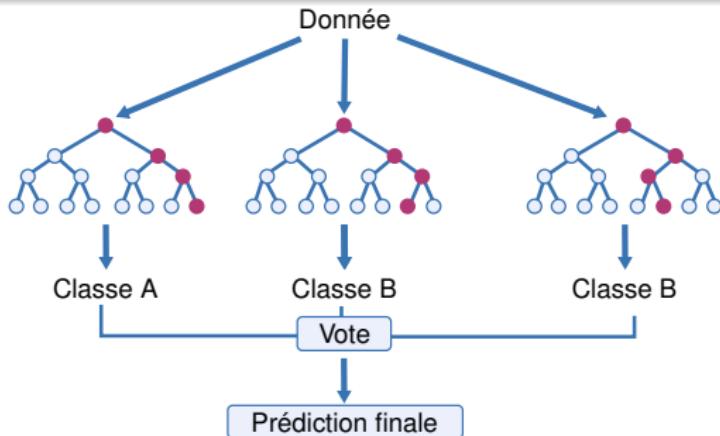
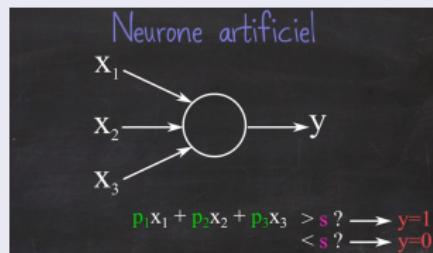


Image inspirée de kdnuggets.com

Exemple de classifieurs

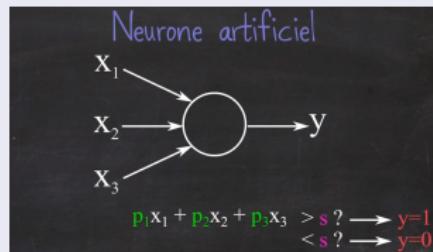
Classifieur - Réseaux de neurones



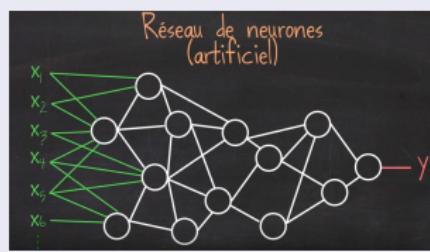
Neurone artificiel

Exemple de classificateurs

Classifieur - Réseaux de neurones



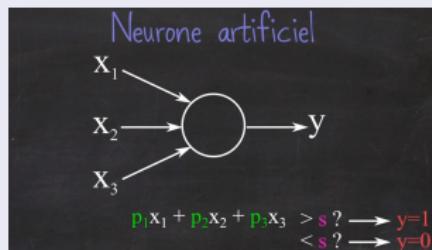
Neurone artificiel



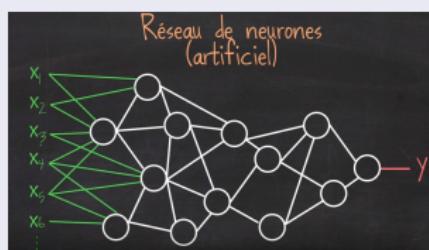
Réseau de neurones

Exemple de classificateurs

Classifieur - Réseaux de neurones



Neurone artificiel



Réseau de neurones

Images issues de



- Lien youtube
- Chaîne : ScienceEtonnante
- Intervenant : David Louapre

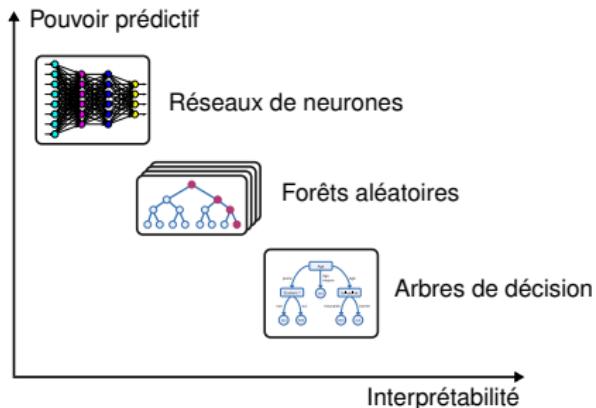
Interprétabilité des classifieurs

Définition - Classifieur interprétable

Peut être appliqué “manuellement” pour obtenir une décision

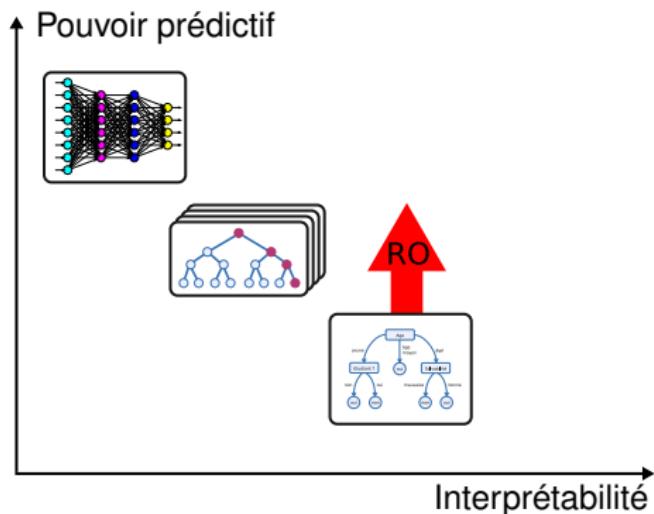
Pourquoi s'intéresser à l'interprétabilité ?

- Souvent nécessaire pour que l'utilisateur fasse confiance aux résultats
Ex : médecins, patients, décideurs, ...
- Permet d'extraire des connaissances
- Parfois imposé par la loi
Right to explanation, credit scoring, ...



Intérêt de la RO

- Permet l'obtention de classifieurs par méthodes exactes
- Améliore le pouvoir prédictif de modèles de classificateurs interprétables



Inconvénients

- Méthodes exactes souvent lentes
- Nécessite des données d'apprentissage de faible taille

Sommaire

1 Introduction au machine learning

2 ORC - Règles ordonnées pour la classification

- Exemple introductif
- Étape 1 - Génération de règles
- Étape 2 - Classement des règles générées
- Résultats

3 Projet

Sommaire

1 Introduction au machine learning

2 ORC - Règles ordonnées pour la classification

- Exemple introductif
- Étape 1 - Génération de règles
- Étape 2 - Classement des règles générées
- Résultats

3 Projet

- Présentation du sujet
- Julia

Définition - Classifieur associatif

Classifieur basé sur des règles d'association :



Ex : si taux d'alcool $\geq 20\%$, je prédis "vin"

Définition - Classifieur associatif de type liste de décision

- Contient une liste ordonnée de règles d'associations
- La 1ère règle vérifiée par une donnée détermine sa classe

Règle n°1 : Taux d'alcool $\geq 20\%$ → Vin

Règle n°2 : Couleur $\in [200, 300]\text{nm}$ → Bière

⋮

⋮

⋮

Règle n°L : Taux d'alcool $\leq 10\%$ → Bière



Règle n°1 : Taux d'alcool $\geq 20\%$ → Vin

Règle n°2 : Couleur $\in [200, 300]\text{nm}$ → Bière

⋮

⋮

⋮

Règle n°L : Taux d'alcool $\leq 10\%$ → Bière



Règle n°1 : Taux d'alcool $\geq 20\%$ → Vin X

Règle n°2 : Couleur $\in [200, 300]\text{nm}$ → Bière

⋮

⋮

⋮

Règle n°L : Taux d'alcool $\leq 10\%$ → Bière



Règle n°1 : Taux d'alcool $\geq 20\%$ → Vin

X

Règle n°2 : Couleur $\in [200, 300]\text{nm}$ → Bière

✓

⋮

⋮

⋮

Règle n°L : Taux d'alcool $\leq 10\%$ → Bière



Règle n°1 : Taux d'alcool $\geq 20\%$ → Vin

X

Règle n°2 : Couleur $\in [200, 300]\text{nm}$ → Bière

✓ Je prédis bière

:

:

:

Règle n°L : Taux d'alcool $\leq 10\%$ → Bière



Règle n°1 : Taux d'alcool $\geq 20\%$ → Vin

X

Règle n°2 : Couleur $\in [200, 300]\text{nm}$ → Bière

✓ Je prédis bière

:

:

:

Règle n°L : Taux d'alcool $\leq 10\%$ → Bière



Règle n°1 : Taux d'alcool $\geq 20\%$ → Vin

Règle n°2 : Couleur $\in [200, 300]\text{nm}$ → Bière

Je prédis bière

⋮

⋮

⋮

Règle n°L : Taux d'alcool $\leq 10\%$ → Bière



Règle n°1 : Taux d'alcool $\geq 20\%$ → Vin

✓ Je prédis vin

Règle n°2 : Couleur $\in [200, 300]\text{nm}$ → Bière

✓ Je prédis bière

:

:

:

Règle n°L : Taux d'alcool $\leq 10\%$ → Bière



- Règle n°1 : Taux d'alcool $\geq 20\%$ → Vin X ✓ Je prédis vin
- Règle n°2 : Couleur $\in [200, 300]\text{nm}$ → Bière ✓ Je prédis bière
- ⋮ ⋮ ⋮
- Règle n°L : Taux d'alcool $\leq 10\%$ → Bière

Méthode - Ordered Rules for Classification (ORC)

Un classifieur associatif de type liste de décision obtenu par résolution de PLNE



Allison Chang, Dimitris Bertsimas, and Cynthia Rudin.

An integer optimization approach to associative classification.

In Advances in neural information processing systems, pages 269–277, 2012.

Avantages

- Performances comparables à celles des méthodes de l'état de l'art
- Simple
- Interprétable

Désavantage

- Résolution exacte lente
- Données sous forme de vecteurs binaires
[One-hot encoding](#)

Couleur $\in [0, 300\text{nm}[$	Couleur $\in [300, 600\text{nm}[$	Couleur $\geq 600\text{nm}$	Alcool $\in [0, 15\%[$	Alcool $\geq 15\%$	Classe
------------------------------------	--------------------------------------	--------------------------------	---------------------------	-----------------------	--------

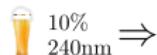


Avantages

- Performances comparables à celles des méthodes de l'état de l'art
- Simple
- Interprétable

Désavantage

- Résolution exacte lente
- Données sous forme de vecteurs binaires
[One-hot encoding](#)



Couleur $\in [0, 300\text{nm}[$	Couleur $\in [300, 600\text{nm}[$	Couleur $\geq 600\text{nm}$	Alcool $\in [0, 15\%[$	Alcool $\geq 15\%$	Classe
1	0	0	1	0	0

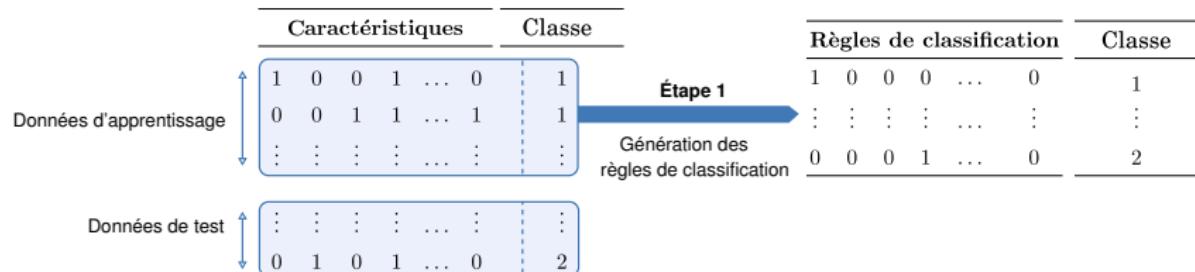
0 si bière
1 si vin

Caractéristiques							Classe
1	0	0	1	...	0		1
0	0	1	1	...	1		1
:	:	:	:	...	:		:
:	:	:	:	...	:		:
0	1	0	1	...	0		2

	Caractéristiques						Classe
Données d'apprentissage	1	0	0	1	...	0	1
	0	0	1	1	...	1	1
	:	:	:	:	...	:	:
Données de test	:	:	:	:	...	:	:
	0	1	0	1	...	0	2

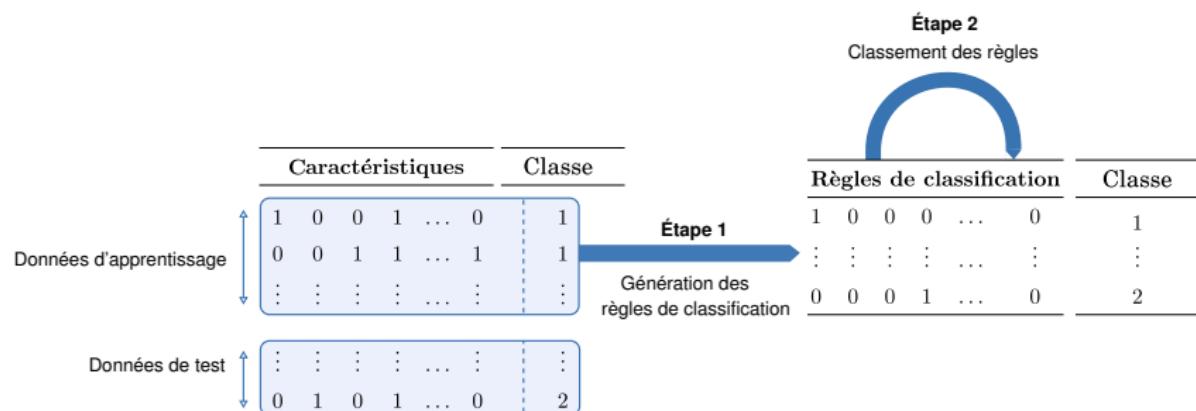
ORC - Principe

1 Extraction de règles associatives



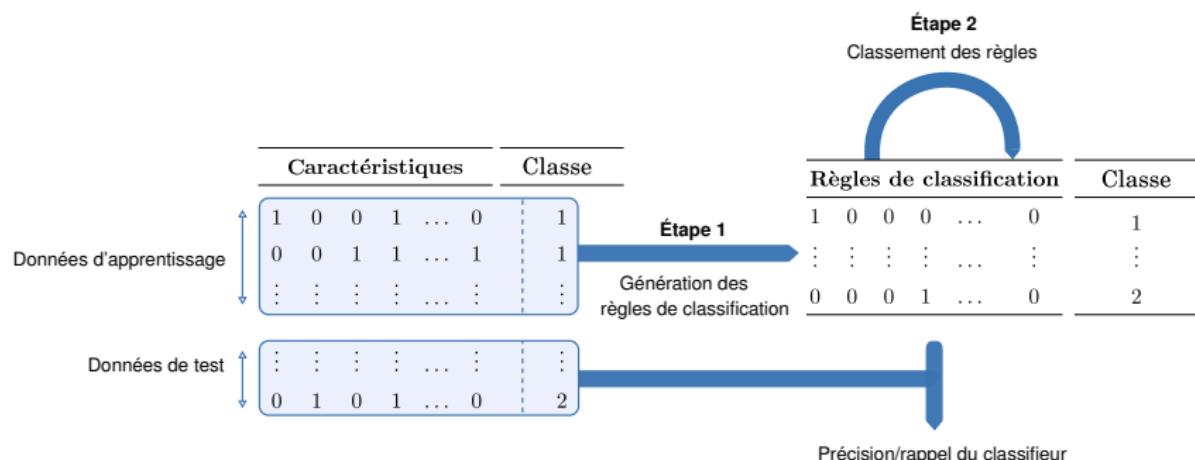
ORC - Principe

- ① Extraction de règles associatives
- ② Classement optimal des règles



ORC - Principe

- ① Extraction de règles associatives
- ② Classement optimal des règles

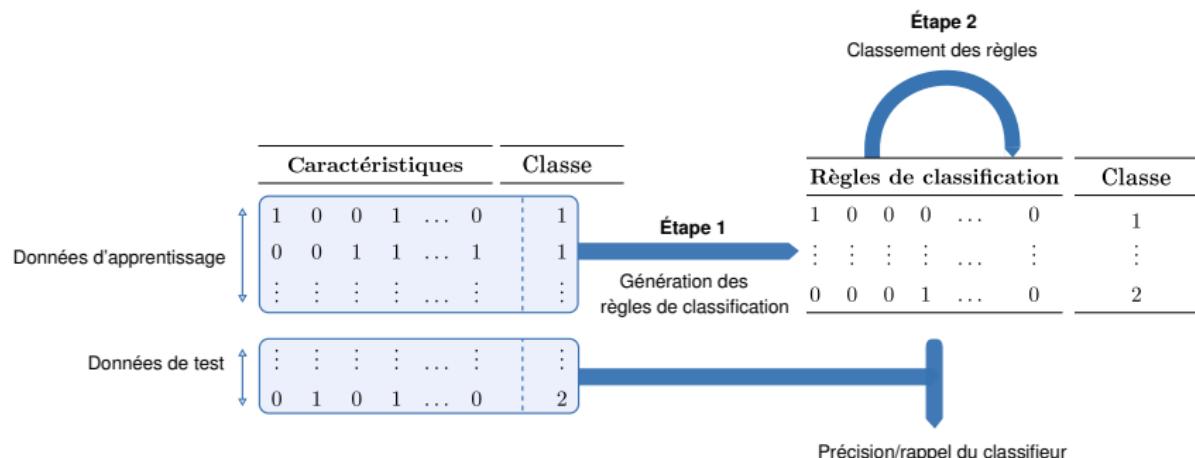


ORC - Principe

- 1 Extraction de règles associatives
- 2 Classement optimal des règles

Trouver et ordonner optimalement des règles sont des problèmes combinatoires

Spécialité de la programmation mixte en nombres entiers



Sommaire

1 Introduction au machine learning

2 ORC - Règles ordonnées pour la classification

- Exemple introductif
- **Étape 1 - Génération de règles**
- Étape 2 - Classement des règles générées
- Résultats

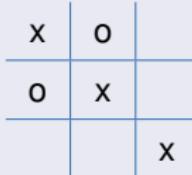
3 Projet

- Présentation du sujet
- Julia

Exemple introductif - Jeu de morpion

Principe

- 2 joueurs x et o
- Tour par tour
- x joue en 1er
- Le 1er joueur alignant 3 de ses symboles gagne



Problème de classification

Étant donnée une grille, déterminer si le joueur x **a gagné** ou non

2 classes

- ① x a gagné
- ② x n'a pas gagné

Représentation des données

Définition - Transaction

Vecteur binaire représentant une donnée

Morpion - Transaction

1 transaction = contenu d'1 grille

Notations

- d : nombre de caractéristiques
- $t \in \{0, 1\}^d$: transaction

Morpion - Caractéristiques

- 9 cases
- 3 valeurs possibles : $\{x, o, \emptyset\}$
- ⇒ 27 caractéristiques

1	2	3
4	5	6
7	8	9

Morpion - Caractéristiques

- 9 cases
- 3 valeurs possibles : $\{x, o, \emptyset\}$
- ⇒ 27 caractéristiques

1	2	3
4	5	6
7	8	9

Morpion - Exemple de transaction

Grille	
o	x
x	o
o	x

		1	2	3	4	5	6	7	8	9
o	1	0	0	0	1	0	1	0	0	
x	0	1	0	1	0	0	0	0	1	
∅	0	0	1	0	0	1	0	1	0	

Morpion - Caractéristiques

- 9 cases
- 3 valeurs possibles : $\{x, o, \emptyset\}$
- ⇒ 27 caractéristiques

1	2	3
4	5	6
7	8	9

Morpion - Exemple de transaction

Grille	
o	x
x	o
o	x

		1	2	3	4	5	6	7	8	9
o	1			1		1				
x		1		1				1		
\emptyset			1			1		1		

Définition - Règle

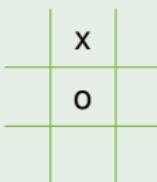
Vecteur binaire de taille d

Morpion - Exemple de règle

- La règle

1	2	3	4	5	6	7	8	9
o								1
x			1					
∅								

- Signifie :
 - la case 2 contient x et
 - la case 5 contient o



Définition - Application

Une règle $b \in \{0, 1\}^d$ s'applique à une transaction $t \in \{0, 1\}^d$ si

$$b \leq t$$

Morpion - Exemple d'application

- La règle

	x	
	o	

	1	2	3	4	5	6	7	8	9
o					1				
x				1					
∅									

- S'applique à la grille

o	x	x	
			o

	1	2	3	4	5	6	7	8	9
o	1				1				1
x			1	1					
∅					1		1	1	1

Objectif du problème de génération **d'une** règle

Trouver une règle qui s'applique

- à beaucoup de transactions d'une classe donnée
- à peu de transactions des autres classes

Données

- b : règle recherchée
- y : classe de b
- S : transactions de classe y

Variables

$$x_i = \begin{cases} 1 & \text{si } b \text{ s'applique à } t_i \\ 0 & \text{sinon} \end{cases} \quad \forall i \in \{1, \dots, n\}$$

Nombre de transactions dans les données d'apprentissage

Définition - Support d'une règle

$$s = \frac{1}{n} \sum_{i \in S} x_i$$

Pourcentage de transactions de classe y auxquelles s'applique b

Données

- b : règle recherchée
- y : classe de b
- S : transactions de classe y

Variables

$$x_i = \begin{cases} 1 & \text{si } b \text{ s'applique à } t_i \\ 0 & \text{sinon} \end{cases} \quad \forall i \in \{1, \dots, n\}$$

Nombre de transactions dans les données d'apprentissage

Définition - Support d'une règle

$$s = \frac{1}{n} \sum_{i \in S} x_i$$

Pourcentage de transactions de classe y auxquelles s'applique b

Définition - Couverture d'une règle de classe y

$$c = \frac{1}{n} \sum_{i=1}^n x_i$$

Pourcentage de transactions auxquelles s'applique b

Problème multi-objectif

Trouver des règles qui s'appliquent

- à beaucoup de transactions de classe y

Problème multi-objectif

Trouver des règles qui s'appliquent

- à beaucoup de transactions de classe y
Maximiser le support

Problème multi-objectif

Trouver des règles qui s'appliquent

- à beaucoup de transactions de classe y
Maximiser le support
- à peu de transactions des autres classes

Problème multi-objectif

Trouver des règles qui s'appliquent

- à beaucoup de transactions de classe y
Maximiser le support
- à peu de transactions des autres classes
Minimiser la couverture

Comment résoudre le problème multi-objectif ?

Option 1

- Maximiser le support

$$\max_{b,x} \sum_{i \in S} x_i$$

- Mettre une borne supérieure c_{max} sur la couverture

Qu'on fera varier

$$\sum_{i=1}^n x_i \leq c_{max}$$

Comment résoudre le problème multi-objectif ?

Option 1

- Maximiser le support

$$\max_{b,x} \sum_{i \in S} x_i$$

- Mettre une borne supérieure c_{max} sur la couverture

Qu'on fera varier

$$\sum_{i=1}^n x_i \leq c_{max}$$

Option 2

Résoudre un problème bi-objectif

Question d'ouverture du projet

Fonction objectif complète

$$\max_{b,x} \underbrace{\sum_{i \in S} x_i}_{\text{Support}} - \underbrace{R_{genX} \sum_{i=1}^n x_i}_{\text{Faible couverture}} - \underbrace{R_{genB} \sum_{j=1}^d b_j}_{\text{Parcimonie}}$$

- $R_{genX} < \frac{1}{n}$
- $R_{genB} < \frac{R_{genX}}{d}$

Génération d'une règle b

Variables

- $x_i : 1$ si b s'applique à t_i 0 sinon
- $b_j : 1$ si b contient la caractéristique j 0 sinon

Contraintes

- (1) $x_i \leq 1 + (t_{ij} - 1)b_j \quad \forall t_i \quad \forall j \leftarrow \text{Caractéristique}$
 - ↑ Transaction
Si $j \in b$ et $j \notin t_i$, alors b ne s'applique pas à t_i
- (2) $x_i \geq 1 + \sum_{j=1}^d (t_{ij} - 1)b_j \quad \forall t_i$
 - ↑ Si $\forall j \in b$ $j \in t_i$, alors b s'applique à t_i
- (3) $\sum_{i=1}^n x_i \leq c_{max} \leftarrow \text{La borne sur la couverture est respectée}$

Génération d'une règle b

Problème de génération d'une règle pour une classe y

$$P(y, c_{max}) = \begin{cases} \max_{b,x} & \sum_{i \in S} x_i - R_{genX} \sum_{i=1}^n x_i - R_{genB} \sum_{j=1}^d b_j \\ \text{s.c.} & (1) \text{ à } (5) \\ & b_j \in \{0, 1\} \forall j \in \{1, \dots, d\} \\ & x_i \in [0, 1] \forall i \in \{1, \dots, n\} \end{cases}$$

Résoudre $P(y, c_{max})$ pour toutes valeurs de c_{max} pertinente

Solutions équivalentes

Pour une valeur de c_{max} donnée :

- il peut exister plusieurs règles optimales
- on souhaite pouvoir en obtenir plusieurs

Solutions équivalentes

Pour une valeur de c_{max} donnée :

- il peut exister plusieurs règles optimales
- on souhaite pouvoir en obtenir plusieurs

Solution choisie

Répéter

- Résoudre $P(y, c_{max})$ pour obtenir
 - la règle b^*
 - le support optimal \bar{s}

Solutions équivalentes

Pour une valeur de c_{max} donnée :

- il peut exister plusieurs règles optimales
- on souhaite pouvoir en obtenir plusieurs

Solution choisie

Répéter

- Résoudre $P(y, c_{max})$ pour obtenir
 - la règle b^*
 - le support optimal \bar{s}
- Ajouter la contrainte

$$\sum_{j : b_j^*=0} b_j + \sum_{j : b_j^*=1} (1 - b_j) \geq 1 \quad (*)$$

Assure de ne plus obtenir la règle b^*

Jusqu'à ce que l'objectif soit $< \bar{s}$

Version haut niveau de l'algorithme

pour chaque classe y **faire**

└ Couverture maximale autorisée

$c_{max} \leftarrow n$

répéter

si c'est la première règle générée avec cette valeur de c_{max} **alors**

└ Support

$\bar{s}, b \leftarrow$ Résoudre $P(y, c_{max}) \leftarrow$ Génération d'une nouvelle règle

└ Règle

Ajouter la règle b

Empêcher de regénérer cette règle \leftarrow Ajout de la contrainte (*)

si le nombre maximal de règles générées avec c_{max} n'est pas atteint **alors**

$sTemp, b \leftarrow$ Résoudre $P(y, c_{max}) \leftarrow$ Génération d'une nouvelle règle

si la nouvelle solution est dominée **alors**

$c_{max} \leftarrow c_{max} - 1$

$sTemp < \bar{s}$

sinon

$c_{max} \leftarrow c_{max} - 1$

jusqu'à $c_{max} = 1$

Entrées : mincov, iter_lim

pour toute classe y faire

Règles générées

$(\mathcal{R}_Y, \bar{s}, iter, c_{max}) \leftarrow (\emptyset, 0, 1, n)$

répéter

si iter = 1 alors

$\bar{s}, b \leftarrow$ Résoudre $P(y, c_{max})$

$iter \leftarrow iter + 1$

$\mathcal{R}_Y \leftarrow \mathcal{R}_Y \cup b$

Ajouter la contrainte (*)

si iter < iter_lim alors

$sTemp, b \leftarrow$ Résoudre $P(y, c_{max})$

si $sTemp < \bar{s}$ alors

$c_{max} \leftarrow \min(c_{max} - 1, \sum_{i=1}^n x_i)$

$iter \leftarrow 1$

sinon

$iter \leftarrow iter + 1$

sinon

$c_{max} \leftarrow c_{max} - 1$

$iter \leftarrow 1$

jusqu'à $c_{max} < n$ mincov

retourner \mathcal{R}_Y

Paramètres

- *mincov* : valeur minimale de c_{max}
- *iter_lim* : nombre maximum de règles générées par $P(y, c_{max})$

Sommaire

1 Introduction au machine learning

2 ORC - Règles ordonnées pour la classification

- Exemple introductif
- Étape 1 - Génération de règles
- Étape 2 - Classement des règles générées**
- Résultats

3 Projet

- Présentation du sujet
- Julia

Comment classifier une donnée/transaction ?

- ① Ordonner les L règles générées
- ② Attribuer à chaque transaction la classe de la 1^{ère} règle qu'elle vérifie



Règle n°1 : Taux d'alcool $\geq 20\%$ → Vin



✓ Je prédis vin

Règle n°2 : Couleur $\in [200, 300]\text{nm}$ → Bière



✓ Je prédis bière

⋮

⋮

⋮

Règle n°L : Taux d'alcool $\leq 10\%$ → Bière

Objectif

Trouver l'ordre qui maximise la reconnaissance des données d'apprentissage

Remarques

- Problème combinatoire
 $L!$ possibilités
- Résolution par PLNE appropriée

Règles nulles (vérifiées par toutes les transactions)

Ajout des règles :

- $\emptyset \Rightarrow 1$
- $\emptyset \Rightarrow -1$

Règles nulles (vérifiées par toutes les transactions)

Ajout des règles :

- $\emptyset \Rightarrow 1$
- $\emptyset \Rightarrow -1$

Données

- $p_{il} = \begin{cases} 0 & \text{si la règle } l \text{ ne s'applique pas à } t_i \\ 1 & \text{si la règle } l \text{ classifie correctement } t_i \\ -1 & \text{si la règle } l \text{ ne classifie pas correctement } t_i \end{cases}$
- $v_{il} = |p_{il}|$

Variables

- r_l : rang de la règle l $\forall l \in \{1, \dots, L\}$
- r_* : rang de la plus haute règle nulle
- $u_{il} = \begin{cases} 1 & \text{si la règle } l \text{ est la plus haute s'appliquant à } t_i \\ 0 & \text{sinon} \end{cases}$

Objectif

$$\max_{r, r_*, u} \sum_{i=1}^n \sum_{l=1}^L p_{il} u_{il} + R_{rank} r_*$$

$\uparrow \leq \frac{1}{L}$ Minimise le nombre de règles du classifieur

- $\sum_{i=1}^n \sum_{l=1}^L p_{il} u_{il}$: transactions bien classifiées - transactions mal classifiées
- $R_{rank} r_*$: rang de la plus haute règle nulle

Parcimonie

Fixation de u_{il}

Variables

- g_i : rang de la plus haute règle s'appliquant à t_i

Contraintes

Pour toute transaction t_i

- $\sum_{l=1}^K u_{il} = 1$
 └ Une règle est associée à t_i

Pour toute transaction t_i et toute règle l

- $g_i \geq v_{il}r_l$
 └ Si la règle s'applique, $g_i \geq r_l$
- $g_i \leq v_{il}r_l + L(1 - u_{il})$
 └ Si la règle est la plus haute qui s'applique alors $g_i \leq r_l$
- $u_{il} \in \{0, 1\}$

Fixation de r_l

Variables

- $s_{lk} = \begin{cases} 1 & \text{si la règle } l \text{ a le rang } k \\ 0 & \text{sinon} \end{cases}$

Contraintes

Pour toute règle r_l

- $\sum_{k=1}^L s_{lk} = 1$
 - La règle a un unique rang
- $r_l = \sum_{k=1}^L k s_{lk}$
 - Si la règle a le rang k , $r_l = k$
- $r_l \in \{1, \dots, L\}$

Pour tout rang k

- $\sum_{l=1}^L s_{kl} = 1$
 - Le rang a une unique règle
- $s_{lk} \in \{0, 1\}$

Fixation de r_*

Variables

- $r_A = r_{L-1}$: rang de $\emptyset \Rightarrow -1$
- $r_B = r_L$: rang de $\emptyset \Rightarrow 1$
- $\alpha = \begin{cases} 1 & \text{si } r_* = r_b \\ 0 & \text{sinon} \end{cases}$
- $\beta = \begin{cases} 1 & \text{si } r_* = r_a \\ 0 & \text{sinon} \end{cases}$

Contraintes

$$r_* \geq r_A$$

$$r_* \geq r_B$$

$$r_* \leq r_A + (L-1)\alpha$$

$$r_* \leq r_B + (L-1)\beta$$

$$r_A \leq r_* + (L-1)\alpha$$

$$r_B \leq r_* + (L-1)\beta$$

$$\alpha + \beta = 1$$

$$\alpha \in \{0, 1\}$$

$$\beta \in [0, 1]$$

Renforcement de formulation

Contraintes optionnelles

$$u_{il} \geq 1 - g_i + v_{il} + r_l \quad \forall i\{1, \dots, n\} \quad \forall l\{1, \dots, L\}$$

$$u_{il} \leq v_{il} \quad \forall i\{1, \dots, n\} \quad \forall l\{1, \dots, L\}$$

$$u_{il} \leq 1 - \frac{r_* - r_l}{L-1} \quad \forall i\{1, \dots, n\} \quad \forall l\{1, \dots, L\}$$

Améliorent la relaxation linéaire

Sommaire

1 Introduction au machine learning

2 ORC - Règles ordonnées pour la classification

- Exemple introductif
- Étape 1 - Génération de règles
- Étape 2 - Classement des règles générées
- **Résultats**

3 Projet

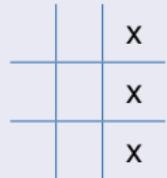
- Présentation du sujet
- Julia

Temps de calcul

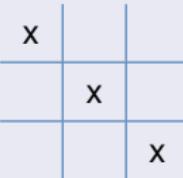
Données	n	d	#Règles	Génération (sec)	Classement (sec)
B.Cancer	683	27	198 ± 16	616 ± 57	12959 ± 1341
CarEval	1728	21	58	706 ± 177	7335 ± 2083
Crime1	426	41	100 ± 15	496 ± 88	12364 ± 7100
Crime2	436	16	27 ± 2	59 ± 30	2546 ± 3450
Haberman	306	10	15 ± 0	14 ± 4	6 ± 2
Mammo	830	25	58 ± 1	670 ± 34	3753 ± 3229
MONK2	432	17	45 ± 4	124 ± 11	5314 ± 2873
SPECT	267	22	145 ± 7	71 ± 9	8862 ± 2292
TicTacToe	958	27	53 ± 3	1241 ± 38	4031 ± 3233
Titanic	2201	8	24 ± 1	92 ± 15	1491 ± 1088

Résultat du jeu de morpion

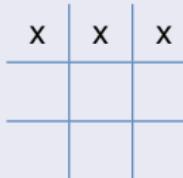
9 règles



1 - x a gagné



2 - x a gagné



3 - x a gagné



4 - x a gagné



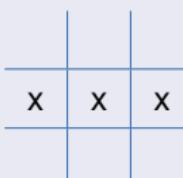
5 - x a gagné



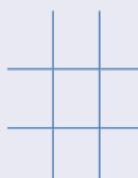
6 - x a gagné



7 - x a gagné



8 - x a gagné



9 - x n'a pas gagné

Performances

Jeu de données		LR	SVM	CART	C4.5	RF	ADA	ORC
B.Cancer	train	0.97	0.98	0.95	0.96	0.98	0.96	0.97
	test	0.95	0.96	0.94	0.95	0.95	0.96	0.95
CarEval	train	0.95	0.98	0.96	0.99	0.99	0.99	0.95
	test	0.94	0.97	0.96	0.98	0.98	0.98	0.95
Crime1	train	0.84	0.84	0.83	0.89	0.99	0.88	0.88
	test	0.73	0.73	0.74	0.74	0.76	0.77	0.78
Crime2	train	0.68	0.74	0.68	0.74	0.82	0.71	0.71
	test	0.67	0.63	0.61	0.59	0.62	0.66	0.66
Haberman	train	0.77	0.78	0.76	0.77	0.78	0.77	0.76
	test	0.75	0.73	0.74	0.73	0.73	0.73	0.75
Mammo	train	0.84	0.86	0.84	0.85	0.88	0.85	0.85
	test	0.83	0.82	0.83	0.83	0.82	0.84	0.83
MONK2	train	0.64	0.67	0.75	0.93	0.99	0.79	0.82
	test	0.60	0.67	0.66	0.88	0.65	0.63	0.73
SPECT	train	0.87	0.86	0.83	0.88	0.93	0.88	0.89
	test	0.79	0.84	0.78	0.79	0.80	0.80	0.77
TicTacToe	train	0.98	0.94	0.93	0.97	1.00	0.99	1.00
	test	0.98	0.91	0.88	0.92	0.97	0.97	1.00
Titanic	train	0.77	0.79	0.78	0.79	0.79	0.78	0.79
	test	0.77	0.78	0.78	0.79	0.78	0.77	0.79

Sommaire

1 Introduction au machine learning

2 ORC - Règles ordonnées pour la classification

3 Projet

- Présentation du sujet
- Julia

Sommaire

1 Introduction au machine learning

2 ORC - Règles ordonnées pour la classification

- Exemple introductif
- Étape 1 - Génération de règles
- Étape 2 - Classement des règles générées
- Résultats

3 Projet

- Présentation du sujet
- Julia

Informations générales

Groupe

- Seul ou en binôme

Langage

- Libre
 - Julia conseillé

Calendrier

- 05/02 : 1h30 de TP
- 12/02 : 3h de TP (présentation de l'avancement)
- 11/03 : date limite de rendu

Données

Diagnostique de maladie chronique du rein chez 189 patients dont on connaît :

- l'âge
- 23 données médicales
pression sanguine, taux d'albuminurie, taux de sodium, ...
- s'il a contracté ou non une maladie liée aux reins
[Classe du patient](#)

Objectif

- Utiliser la méthode ORC sur $\frac{2}{3}$ des patients pour obtenir un classifieur
- Évaluer ses performances sur les patients restants

caractéristiques					classe
Âge	Albuminurie	Sodium	...	Malade ?	
48	4	0	...	notckd	
:	:	:	:	:	
57	0	0	...	ckd	

Étapes

	caractéristiques				classe
	Âge	Albuminurie	Sodium	...	
Données initiales 189	48	4	1		<i>ckd</i>
	53	2	3		<i>ckd</i>
	63	3	0		<i>ckd</i>
	:	:	:		:
	58	0	2		<i>notckd</i>

d

Étapes

	caractéristiques					classe
	Âge	Albuminurie	Sodium	...	Malade ?	
Données initiales 189	48	4	1			<i>ckd</i>
	53	2	3			<i>ckd</i>
	63	3	0			<i>ckd</i>
	:	:	:			:
	58	0	2			<i>notckd</i>

Étape 1 Reformulation des caractéristiques
en vecteurs binaires

Caractéristiques						Malade ?
1	0	0	1	...	0	1
0	0	1	1	...	1	1
:	:	:	:	...	:	:
:	:	:	:	...	:	:
0	1	0	1	...	0	2

\longleftrightarrow
 d

Étapes

	caractéristiques				classe
	Âge	Albuminurie	Sodium	...	Malade ?
Données initiales 189	48	4	1		<i>ckd</i>
	53	2	3		<i>ckd</i>
	63	3	0		<i>ckd</i>
	:	:	:		:
	58	0	2		<i>notckd</i>

Étape 1

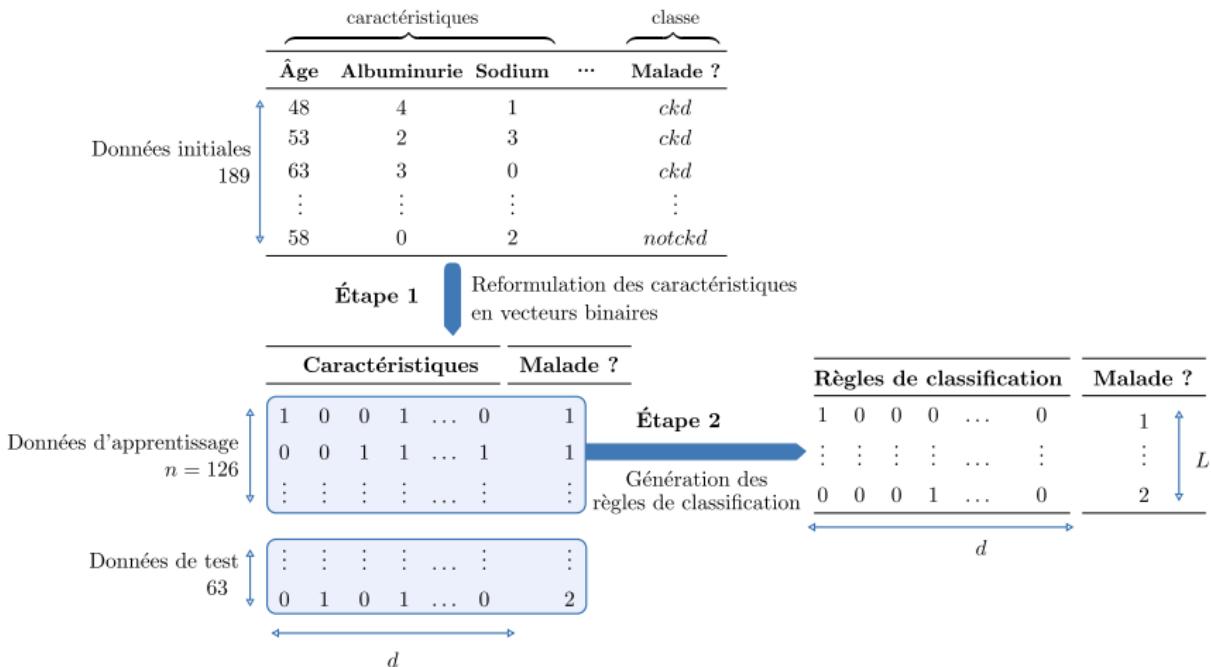
Reformulation des caractéristiques
en vecteurs binaires

	Caractéristiques						Malade ?
Données d'apprentissage $n = 126$	1	0	0	1	...	0	1
	0	0	1	1	...	1	1
	:	:	:	:	...	:	:

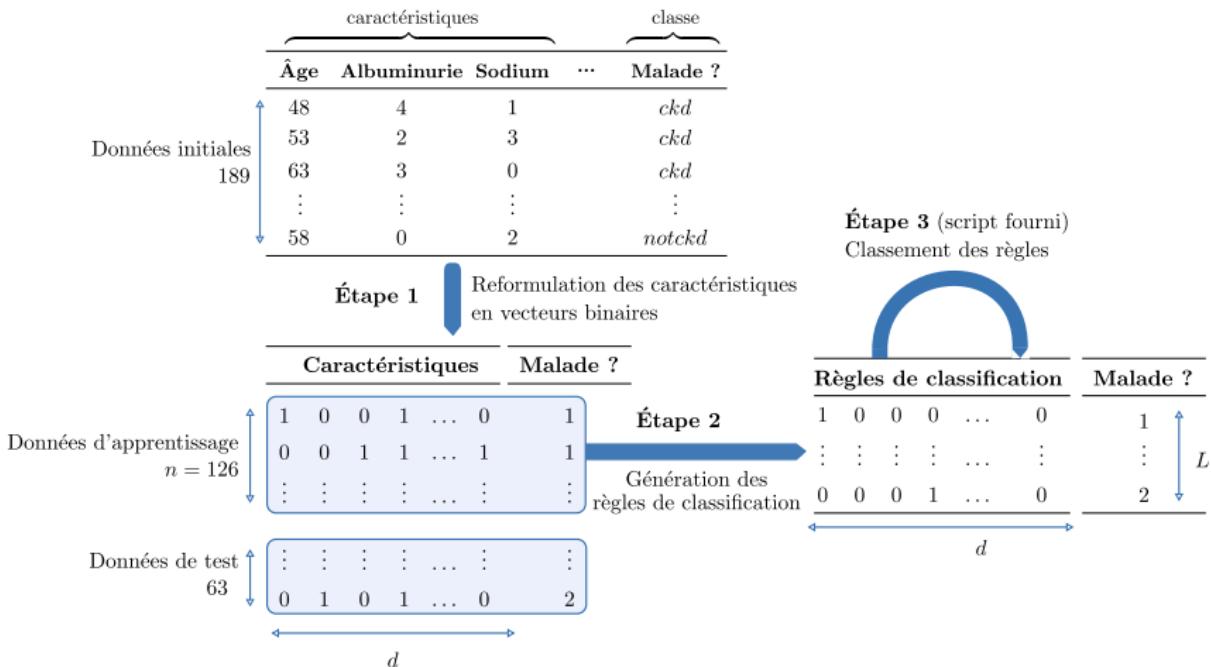
Données de test 63	:	:	:	:	...	:	:
	0	1	0	1	...	0	2

d

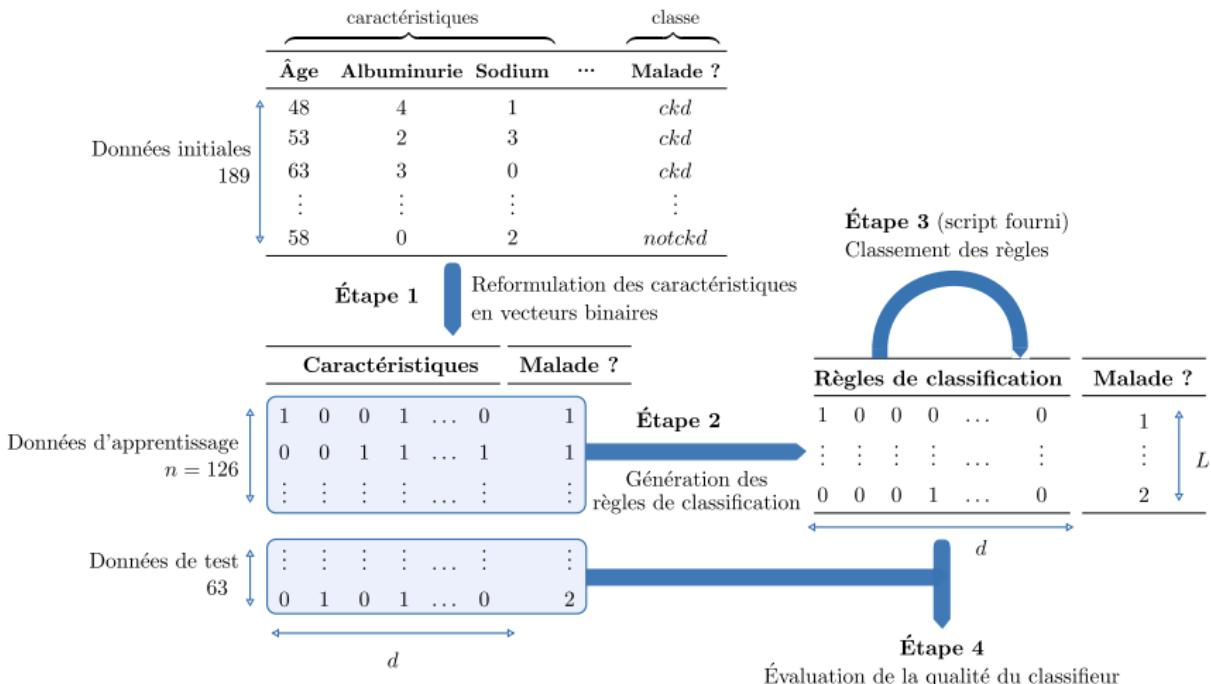
Étapes



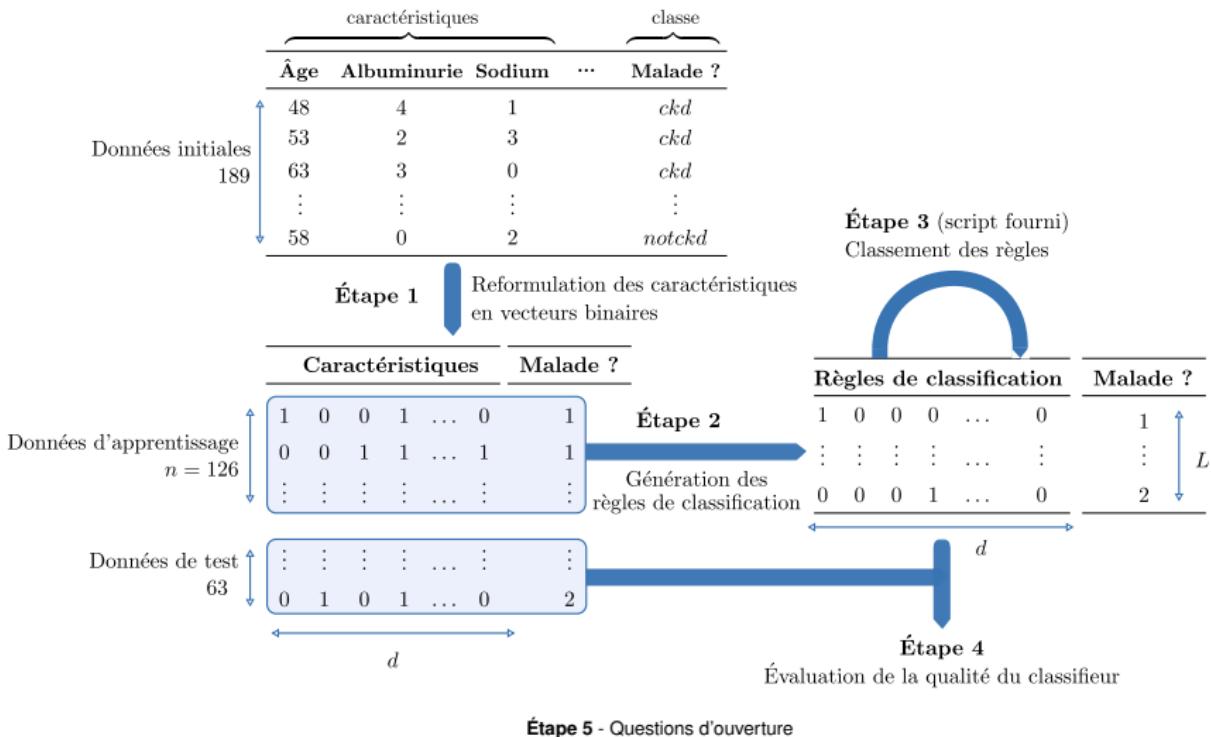
Étapes



Étapes



Étapes



Structure de votre projet

Racine

- data (fichiers de données)
- doc (documents de cours et rapport au format pdf)
- res (règles produites)
- src (code)

Rendu

- Vos fichiers

Fichiers de données, code, résultats

- Votre rapport

- Représentation binaire
- Règles ordonnées du classifieur
- Résultats

Temps, nombre de règles obtenues, performances de votre classifieur, ...

- Questions d'ouverture

Sommaire

1 Introduction au machine learning

2 ORC - Règles ordonnées pour la classification

- Exemple introductif
- Étape 1 - Génération de règles
- Étape 2 - Classement des règles générées
- Résultats

3 Projet

- Présentation du sujet
- Julia

Langage Julia

Avantages

- Performant
Comparable au C++
- Syntaxe simple et efficace
- De plus en plus répandu
Surtout dans la communauté académique
- Facilité de développement et d'utilisation de packages

Package JuMP

Package de Julia permettant de résoudre des problèmes d'optimisation

- Mêmes avantages que Julia
Performant, syntaxe aisée
- Indépendant du solveur
Simple de passer de l'un à l'autre

Déclarer une variable

```
n = 10 # entier  
b = "Hello world" # chaîne de caractères  
v = [1 2 3 4] # vecteur  
m = [1 2; 3 4] # matrice 2x2
```

Inclure un fichier contenant des variables

```
include("monFichier.dat")
```

Affichage

```
println("Afficher du texte")  
println("Afficher une variable $a")  
println("ou ", a)
```

Écrire dans un fichier

```
fout = open("monFichierDeSortie.dat", "a")  
print(fout, v)  
# Remarque :  
# Remplacer "a" par "w" pour écraser l'ancien contenu du fichier
```

Conditionnelle

```
if v[1] == 1
    # contenu du if
else
    # contenu du else
end
```

Boucle for

```
for i in 1:10 # ou i = 1:10
    print(i)
end
```

Boucle while

```
while v[1] == 1
    # contenu de la boucle
end
```

Déclarer un problème d'optimisation avec CPLEX

```
using JuMP
using CPLEX
m = Model(with_optimizer(CPLEX.Optimizer))
# temps limite 300s
set_parameter(m, "CPX_PARAM_TILIM", 300)
```

Déclarer des variables d'un problème d'optimisation

```
# Variable continue
@variable(m, 0 <= x1 <= 1)

# Variable binaire
@variable(m, x2, Bin)

# Tableau n*1
@variable(m, 0 <= y[i in 1:n] <= 1)

# Tableau n*4
@variable(m, 0 <= t[i in 1:n, j in 1:4] <= 1)
```

Définir des contraintes

```
#  $x_1 + x_2 = 1$ 
@constraint(m, x1 + x2 == 1)

#  $y_i + x_1 \leq 1 \quad \forall i \{1, \dots, n\}$ 
@constraint(m, [i = 1:n], y[i] + x1 <= 1)

#  $t_{ij} + x_1 \geq 1 \quad \forall i \{1, \dots, n\} \quad \forall j \{1, \dots, 4\}$ 
@constraint(m, [i = 1:n, j = 1:4], t[i, j] + x1 >= 1)

#  $\sum_{i=1}^n y_i \geq 3$ 
@constraint(m, sum(y[i] >= 3 for i in 1:n))
```

Définir l'objectif

```
@objective(m, Max, sum(y[i] for i = 1:n))

# objectif avec condition
@objective(m, Max, sum(y[i] for i = 1:n if v[i] == 2))
```

Résoudre un problème

```
optimize!(m)
```

Obtenir la valeur d'une variable x1

```
vx1 = JuMP.value(x1)  
vx1Int = trunc(Int, JuMP.value(x1))
```

Obtenir la valeur d'un tableau de variables tx

```
vtx = JuMP.value.(tx)  
vtxInt = trunc(Int, JuMP.value.(tx))
```

Masquer les sorties de CPLEX

```
set_parameter(m, "CPX_PARAM_SCRIND", 0)
```

Problème de sac à dos

Fichier knapsack.jl

```
using JuMP
using CPLEX

include("donnees.dat")

m = Model(with_optimizer(CPLEX.Optimizer))

@variable(m, x[i in 1:n], Bin)
@constraint(m, sum(x[i] * w[i] for i = 1:n) <= K)
@objective(m, Max, sum(x[i] * p[i] for i in 1:n))
optimize!(m)
```

Fichier donnees.dat

```
n = 6
K = 23
w = [1 2 4 5 7 10]
p = [1 3 5 7 9 11]
```

Éxécuter ce fichier à l'ENSTA

- 1 Ouvrir une console : Alt + F2, puis entrer "xterm"

- 2 Fixer les chemins :
usediam ro

- 3 Ajouter les packages nécessaires : julia

```
using Pkg
Pkg.add("JuMP")
Pkg.add("CPLEX")
Pkg.add("CSV")
Pkg.add("DataFrames")
```

- 4 Éxécuter le programme :

```
julia knapsack.jl
# ou si étes en mode console
julia > knapsack.jl
```

Deux types d'exécutions

1 - Commande julia

```
login@pc $ julia knapsack.jl
```

2 - Mode console

```
login@pc $ julia
julia> include("knapsack.jl")
```

↓ Lance le mode console
↑ Exécute le fichier

Avantages du mode console

- Plus pratique pour tester des commandes
- Librairies chargées une seule fois
Sinon prend plusieurs secondes à chaque exécutions

Désavantages du mode console

- Doit être relancé en cas de redéfinition d'une fonction
- Potentiels effets indésirables si variables fixées avant d'inclure un fichier

Package DataFrames

Facilite la manipulation de données d'individus

1 individu par ligne

1 caractéristique par colonne

Utilisation

```
# Lire un fichier csv contenant des entêtes
t = CSV.read("monFichier.csv", header=true)

# Afficher la colonne dont l'entête est age
print(t.age)

# Crée une nouvelle table
tBinaire = DateFrames.DataFrame()

## Binariser une colonne contenant des données numériques
# Ajoute une colonne à tBinaire contenant 1 si l'âge de l'individu est
# <= 15 et 0 sinon
tBinaire.age = ifelse.(t.age .<= 15, 1, 0)

## Binariser une colonne contenant des données catégorielles
# Ajoute une colonne à tBinaire contenant 1 si l'individu est une femme
# et 0 sinon
tBinaire.sex = ifelse.(t.sex .== "female" 1, 0)
```