Une approche d'optimisation discrète pour la classification associative

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

Sommaire

Intervenants

- Axel Parmentier (2 séances)
  Machine learning, génération de colonnes
- Fabien Tarissan(2 séances + 2 pour masters)
  Statistiques, grands graphes
- Zacharie Ales (2 séances)
  PLNE, classification

Planning

Date	Intervenant	CM	TP	Examen
17/01	Z. A.	x	x	
24/01	F. T.	x		
31/01	A. P.	x		
7/02	A. P.	x		
14/02	F. T.	x	x	
21/02	Z. A.		x	x
7/03	F. T.	x	x	
21/03	F. T.		x	

Évaluation

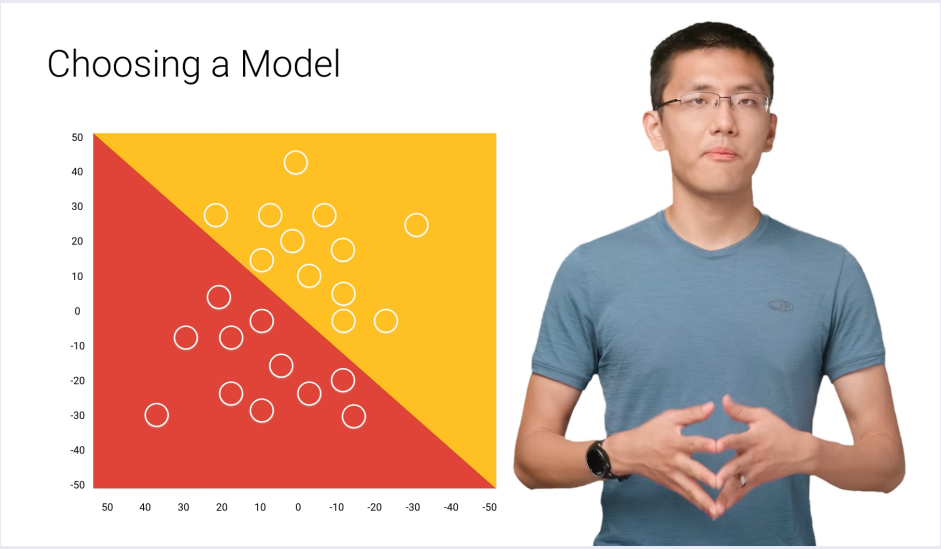
En fonction de l'intervenant

- A. P. : examen
- Z. A. : projet

Sommaire

Vidéo introductive

The 7 Steps of Machine Learning



- Lien youtube
- Chaîne : Google Cloud
- Intervenant : Yufeng Guo

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

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

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

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 - Apprentissage automatique ("machine learning")

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

Définition - Classification

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, ...

Intelligence artificielle $\subset$ Machine learning $\subset$ Classification

Définition - Apprentissage automatique ("machine learning")

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

Définition - Classification

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, ...

Intelligence artificielle $\subset$ Machine learning $\subset$ Classification

Définition - Classifieur

Algorithme de classification
Appelé "model" dans la vidéo

Points importants pour la suite

Partage des données

1. **apprentissage** ("train") : données utilisées pour définir le classifieur
2. **test** : données utilisées pour évaluer les performances du classifieur

Points importants pour la suite

Partage des données

1. **apprentissage** ("train") : données utilisées pour définir le classifieur
2. **test** : données utilisées pour évaluer les performances du classifieur

Définition - **Précision**

Pourcentage de données de test correctement classifiées

Définition - **Rappel**

Pourcentage de données d'apprentissage correctement classifiées

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 ?

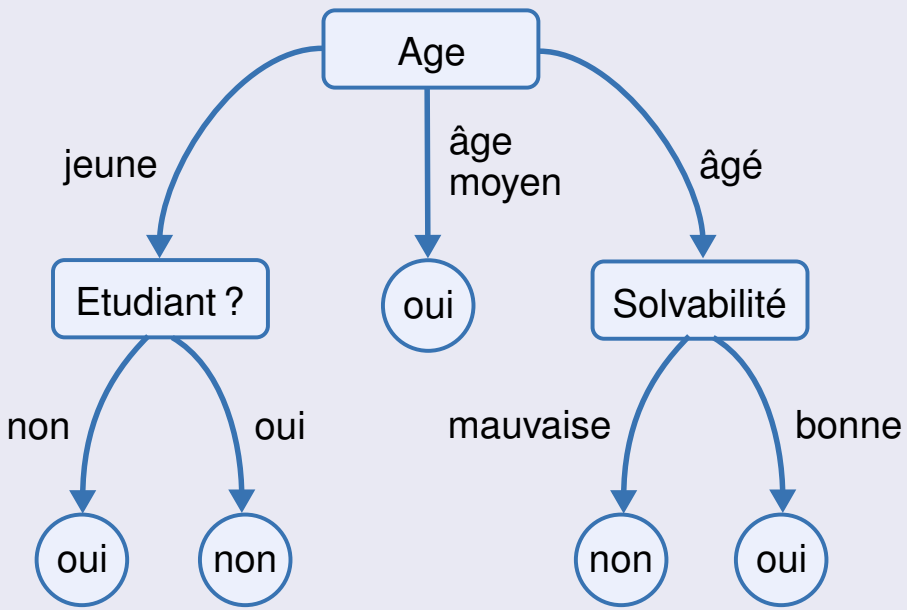


Image inspirée de tutorialspoint.com

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)

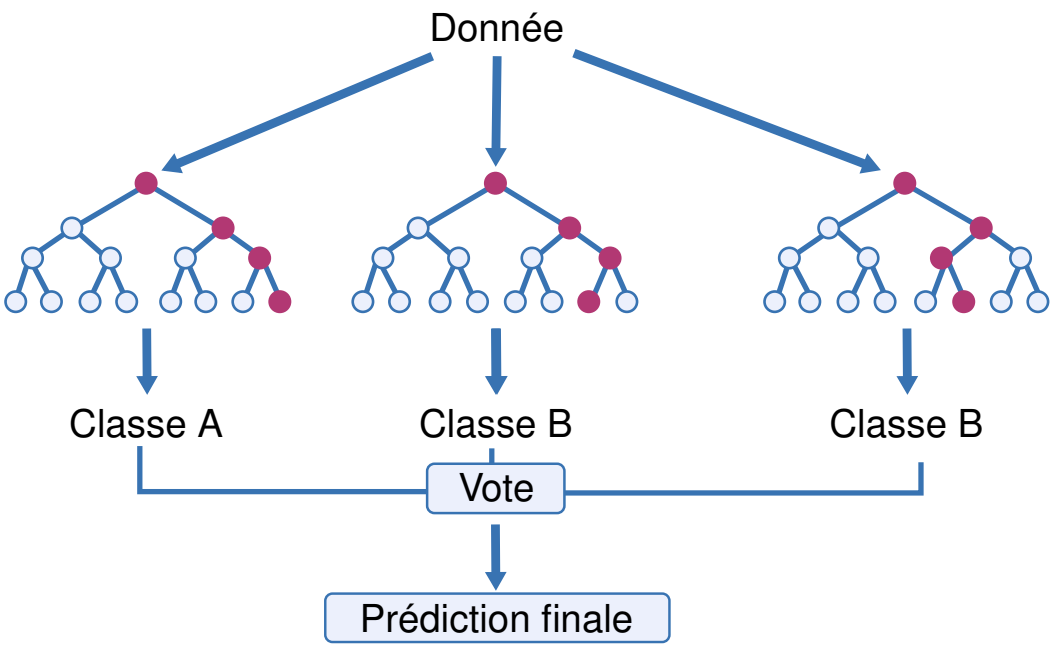
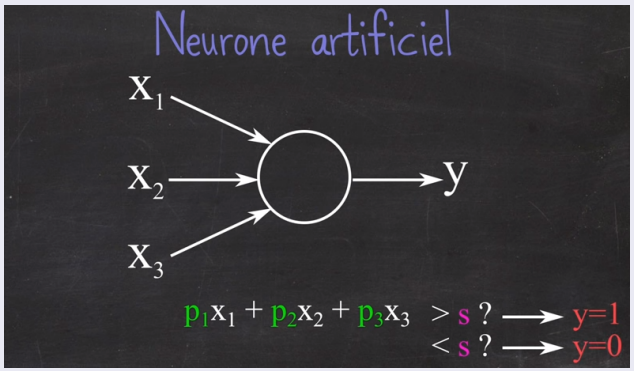


Image inspirée de kdnuggets.com

Exemple de classifieurs

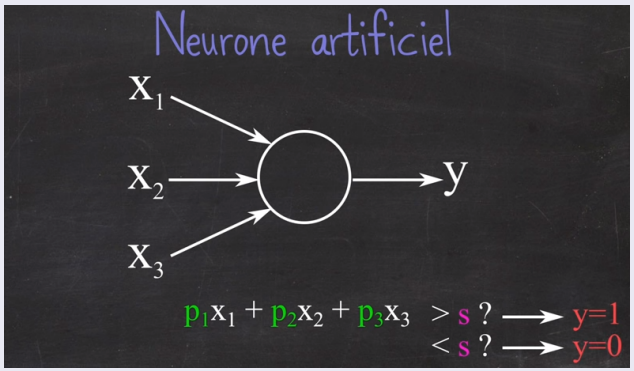
Classifieur - **Réseaux de neurones**



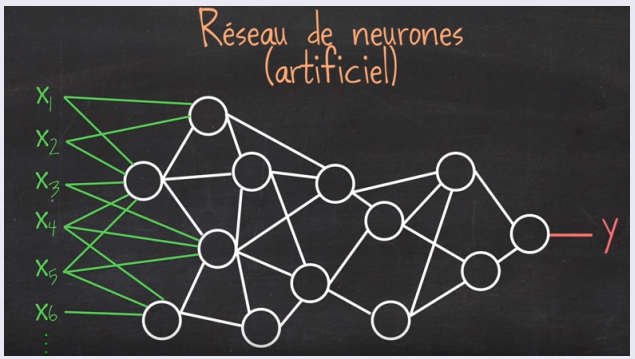
Neurone artificiel

Exemple de classifieurs

Classifieur - **Réseaux de neurones**



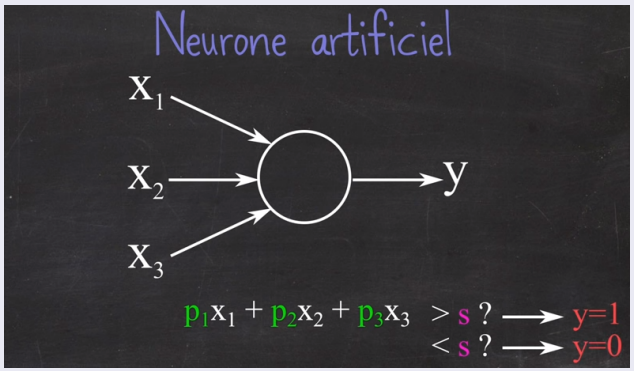
Neurone artificiel



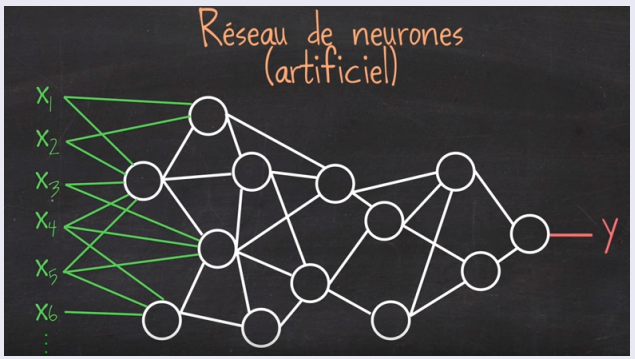
Réseau de neurones

Exemple de classifieurs

Classifieur - **Réseaux de neurones**

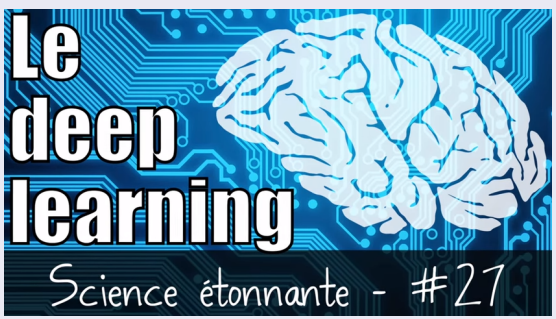


Neurone artificiel



Réseau de neurones

Images issues de



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

Sommaire

Sommaire

Exemple - Règle d'association

Si quelqu'un achète

- $X = \{\text{avocat}, \text{piment}\}$

il a des chances d'acheter également :

- $Y = \{\text{oignon}\}$

Définition - **Règle d'association**

$$X \rightarrow Y$$

avec

- $I = \{i_1, ..., i_m\}$: ensemble d'items
- $X, Y \subseteq I$

Exemple - Règle d'association

Si quelqu'un achète

- $X = \{\text{avocat}, \text{piment}\}$

il a des chances d'acheter également :

- $Y = \{\text{oignon}\}$

Définition - **Règle d'association**

$$X \rightarrow Y$$

avec

- $I = \{i_1, ..., i_m\}$: ensemble d'items
- $X, Y \subseteq I$

Définition - **Classification associative**

- Classifieur basé sur des régles d'association

Dans ce contexte :

- X correspond à des caractéristiques
- Y correspond à une classe

Méthode - **Ordered Rules for Classification** (ORC)

Classifieur associatif de type liste de décision

La classe d'une donnée sera celle de la 1ère règle qu'elle vérifie dans une liste ordonnée de règles

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.

Méthode - **Ordered Rules for Classification** (ORC)

Classifieur associatif de type liste de décision

La classe d'une donnée sera celle de la 1ère règle qu'elle vérifie dans une liste ordonnée de règles

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

Méthode - **Ordered Rules for Classification** (ORC)

Classifieur associatif de type liste de décision

La classe d'une donnée sera celle de la 1ère règle qu'elle vérifie dans une liste ordonnée de règles

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

Définition - **Interprétabilité** d'un classifieur

Capacité à comprendre les décisions prises par un classifieur

Méthode - **Ordered Rules for Classification** (ORC)

Classifieur associatif de type liste de décision

La classe d'une donnée sera celle de la 1ère règle qu'elle vérifie dans une liste ordonnée de règles

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

Définition - **Interprétabilité** d'un classifieur

Capacité à comprendre les décisions prises par un classifieur

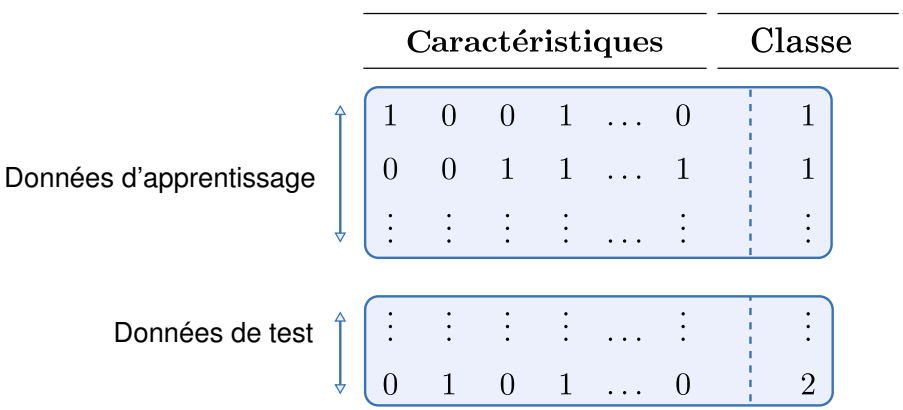
Dichotomie dans les modèles de classification

- Réseaux de neurones : efficaces mais peu interprétables
- Arbres de décision : interprétables mais peu efficaces

ORC - Principe

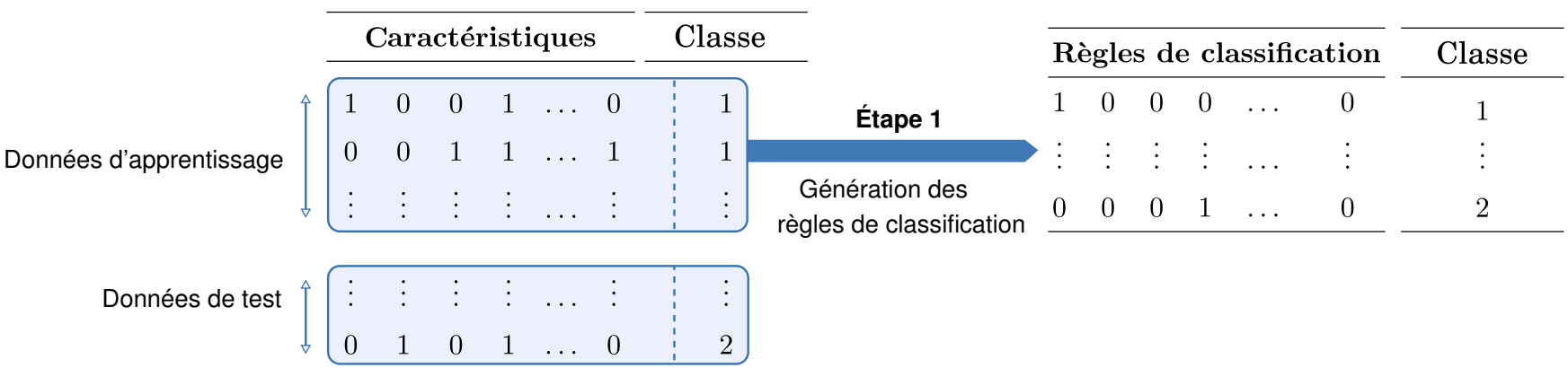
Caractéristiques						Classe
1	0	0	1	...	0	1
0	0	1	1	...	1	1
⋮	⋮	⋮	⋮	...	⋮	⋮
⋮	⋮	⋮	⋮	...	⋮	⋮
0	1	0	1	...	0	2

ORC - Principe



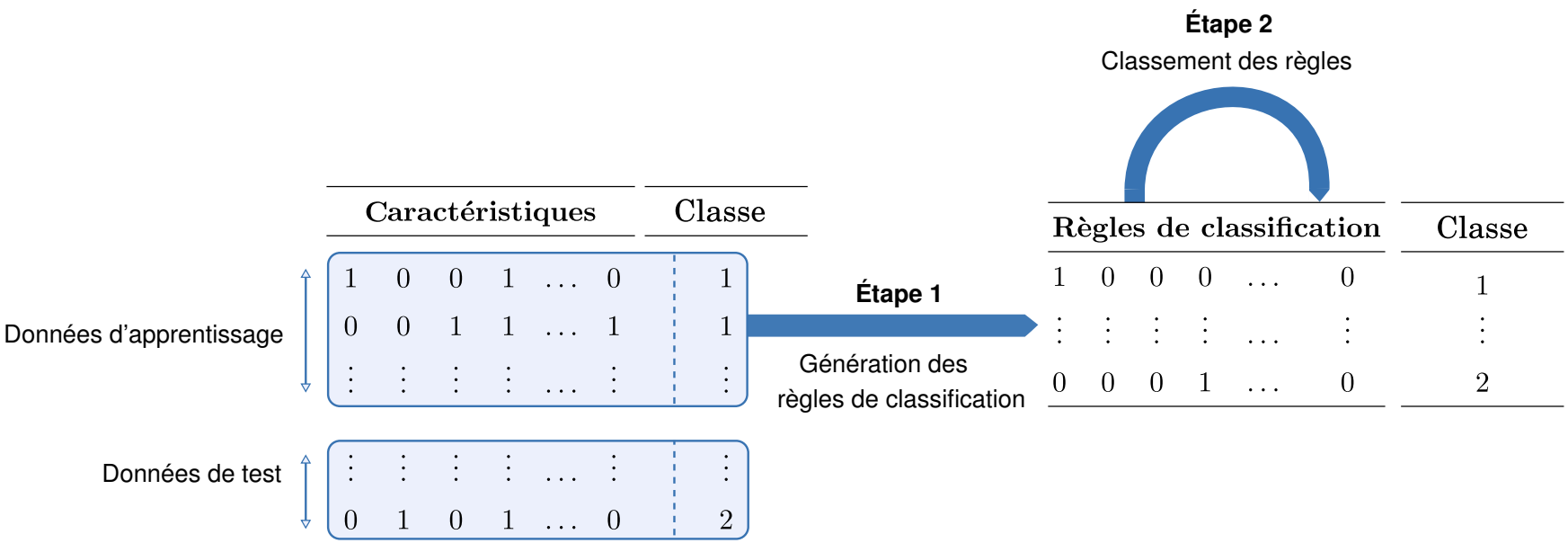
ORC - Principe

1. Extraction de règles associatives



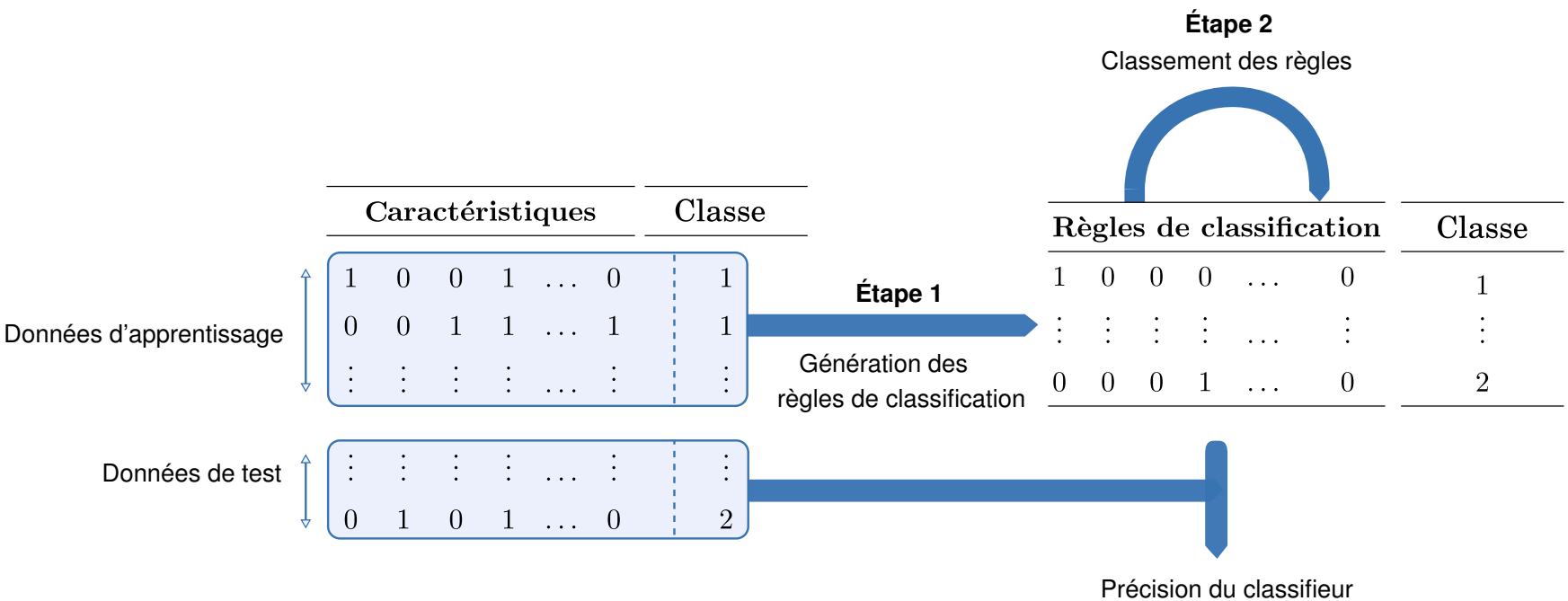
ORC - Principe

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



ORC - Principe

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

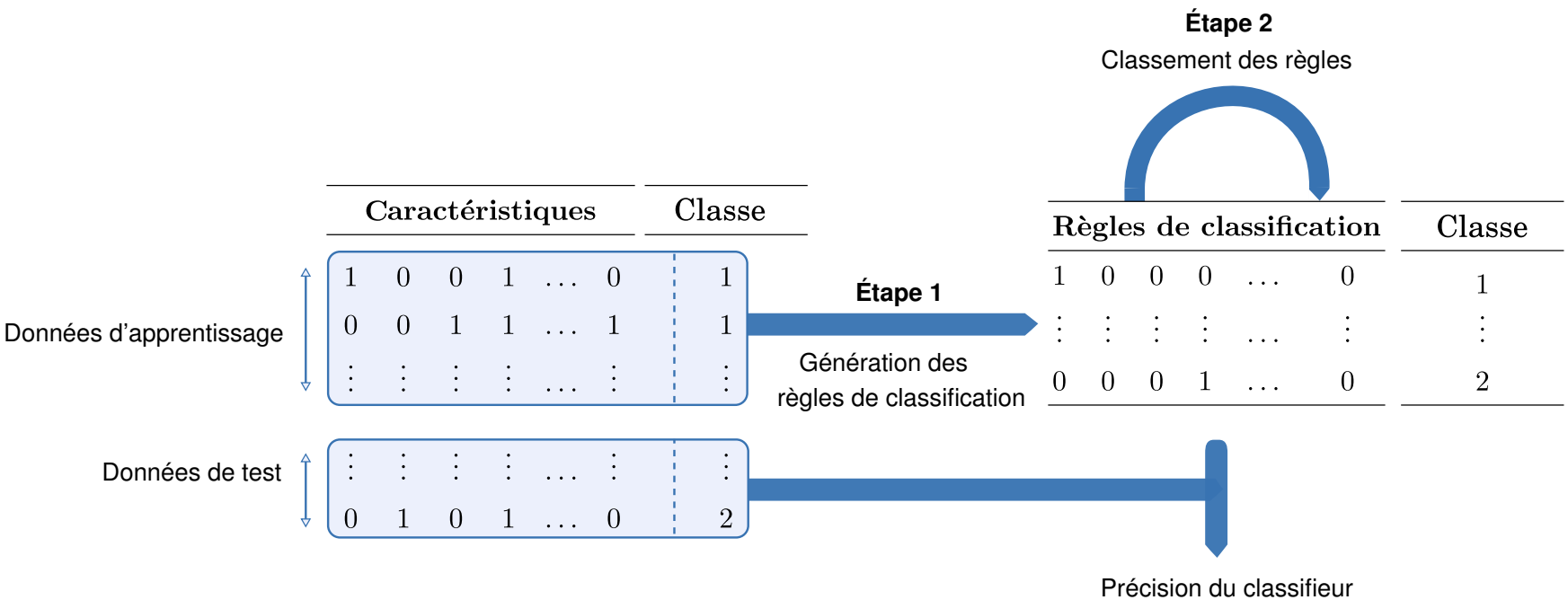


ORC - Principe

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

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

x	o	
o	x	
		x

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

Notations

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

Morpion - Caractéristiques

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

$\Rightarrow$ 27 caractéristiques

1	2	3
4	5	6
7	8	9

Notations

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

Morpion - Caractéristiques

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

$\Rightarrow$ 27 caractéristiques

1	2	3
4	5	6
7	8	9

Morpion - Exemple de transaction

Grille

o	x	
x	o	
o		x

Transaction associée

	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
$\emptyset$	0	0	1	0	0	1	0	1	0

Notations

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

Morpion - Caractéristiques

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

$\Rightarrow$ 27 caractéristiques

1	2	3
4	5	6
7	8	9

Morpion - Exemple de transaction

Grille

o	x	
x	o	
o		x

Transaction associée

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

Définition - **Règle**

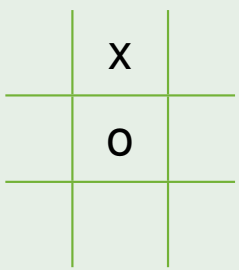
Vecteur binaire de taille d

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



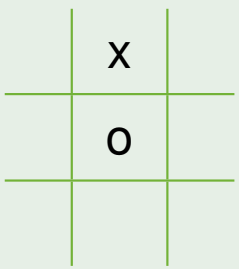
Définition - **Application**

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

$$b \leq t$$

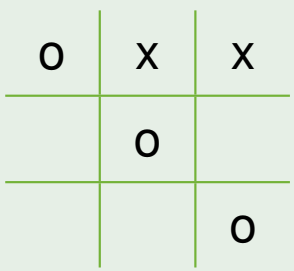
Morpion - Exemple d'application

- La règle



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

- S'applique à la grille



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

Objectif

Trouver une règle *b* qui s'applique

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

Notation

- y : classe de la règle b
- n : nombre de transactions dans les données d'apprentissage
- $x_i = \begin{cases} 1 & \text{si } b \text{ s'applique à } t_i \\ 0 & \text{sinon} \end{cases} \qquad \forall i \in \{1, .., n\}$
- S : ensemble des transactions de classe y

Définition - **Couverture** d'une règle

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

Pourcentage de transactions auxquelles s'applique b

Notation

- y : classe de la règle b
- n : nombre de transactions dans les données d'apprentissage
- $x_i = \begin{cases} 1 & \text{si } b \text{ s'applique à } t_i \\ 0 & \text{sinon} \end{cases} \qquad \forall i \in \{1, .., n\}$
- S : ensemble des transactions de classe y

Définition - **Couverture** d'une règle

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

Pourcentage de transactions auxquelles s'applique b

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

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

Pourcentage de transactions auxquelles s'applique b

Définition - Support

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

Pourcentage de transactions de classe y auxquelles s'applique b

Objectif

Trouver des règles qui s'appliquent

- à beaucoup de transactions de classe y

Définition - Couverture d'une règle

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

Pourcentage de transactions auxquelles s'applique b

Définition - Support

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

Pourcentage de transactions de classe y auxquelles s'applique b

Objectif

Trouver des règles qui s'appliquent

- à beaucoup de transactions de classe y
  Maximiser le support

Définition - Couverture d'une règle

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

Pourcentage de transactions auxquelles s'applique b

Définition - Support

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

Pourcentage de transactions de classe y auxquelles s'applique b

Objectif

Trouver des règles qui s'appliquent

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

Définition - Couverture d'une règle

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

Pourcentage de transactions auxquelles s'applique b

Définition - Support

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

Pourcentage de transactions de classe y auxquelles s'applique b

Objectif

Trouver des règles qui s'appliquent

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

Objectif

Trouver des règles qui s'appliquent :

- à beaucoup de transactions d'une classe donnée
  Maximiser le support
- a peu de transactions des autres classes
  Minimiser la couverture

Option 1

- Maximiser le support
- Mettre une borne supérieure sur la couverture
  Qu'on fera varier

Objectif

Trouver des règles qui s'appliquent :

- à beaucoup de transactions d'une classe donnée
  Maximiser le support
- a peu de transactions des autres classes
  Minimiser la couverture

Option 1

- Maximiser le support
- Mettre une borne supérieure sur la couverture
  Qu'on fera varier

Option 2

Résoudre un problème bi-objectif

Question d'ouverture du projet

Option 1

- Maximiser le support

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

- Mettre une borne supérieure $\overline{s}_X$ sur la couverture

  Qu'on fera varier

$$\sum_{i=1}^{n} x_i \leq \overline{s}_X$$

Fonction objectif complète

$$\max_{b,x} \sum_{i \in S} x_i - R_{genX} \sum_{i=1}^{n} x_i - R_{genB} \sum_{j=1}^{d} b_j$$

- $-R_{genX} \sum_{i=1}^{n} x_i$: faible couverture
- $-R_{genB} \sum_{j=1}^{d} b_j$: parcimonie
- $R_{genX} < \frac{1}{n}$
- $R_{genB} < \frac{R_{genX}}{d}$

Génération d'une règle b

Paramètres

- R_{genX}
- R_{genB}
- $\overline{s}_X$

Variables

- x_i : la règle s'applique à la transaction i
- b_j : la règle contient la caractéristique j

$$P_{\overline{s}_X} = \begin{cases} \max_{b,x} & \sum_{i \in S} x_i - R_{genX} \sum_{i=1}^{n} x_i - R_{genB} \sum_{j=1}^{d} b_j \\ \text{tel que} & \sum_{i=1}^{n} x_i \leq \overline{s}_X \\ & x_i \leq 1 + (t_{ij} - 1) b_j \qquad \forall i \in \{1, .., n\} \; \forall j \in \{1, .., d\} \\ & x_i \geq 1 + \sum_{j=1}^{d} (t_{ij} - 1) b_j \qquad \forall i \in \{1, .., n\} \; \forall j \in \{1, .., d\} \\ & b_j \in \{0, 1\} \qquad \forall j \in \{1, .., d\} \\ & x_i \in [0, 1] \qquad \forall i \in \{1, .., n\} \end{cases}$$

Résoudre $P_{\overline{s}_X}$ pour toutes valeurs de $\overline{s}_X$ pertinente

Solutions équivalentes

Il peut exister plusieurs règles b maximisant l'expression

$$\sum_{i \in S} x_i - R_{genX} \sum_{i=1}^{n} x_i - R_{genB} \sum_{j=1}^{d} b_j$$

On souhaite toutes les obtenir

Solutions équivalentes

Il peut exister plusieurs règles b maximisant l'expression

$$\sum_{i \in S} x_i - R_{genX} \sum_{i=1}^{n} x_i - R_{genB} \sum_{j=1}^{d} b_j$$

On souhaite toutes les obtenir

Solution choisie

Répéter

- Résoudre $P_{\overline{s}_X}$ pour obtenir
  - la règle b^*
  - l'objectif $\overline{s}$

Solutions équivalentes

Il peut exister plusieurs règles b maximisant l'expression

$$\sum_{i \in S} x_i - R_{genX} \sum_{i=1}^{n} x_i - R_{genB} \sum_{j=1}^{d} b_j$$

On souhaite toutes les obtenir

Solution choisie

Répéter

- Résoudre $P_{\overline{s}_X}$ pour obtenir
  - la règle b^*
  - l'objectif $\overline{s}$
- Ajouter la contrainte

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

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

Paramètres

- *mincov* : valeur minimale de $\overline{s}_X$
- *iter_lim* : nombre maximum de règles générées par $P_{\overline{s}_X}$

Notations

- $\overline{s}$: valeur de l'objectif du dernier $P_{\overline{s}_X}$ résolu
- $\mathcal{R}_Y$: ensemble des règles générées

Entrées : mincov, iter_lim
pour toute classe y **faire**
  $(\mathcal{R}_Y, \overline{s}, iter, \overline{s}_X) \leftarrow (\emptyset, 0, 1, n)$
  répéter
    si iter = 1 **alors**
      Résoudre $P_{\overline{s}_X}$
      $\overline{s} \leftarrow \sum_{i \in S} x_i$
      $iter \leftarrow iter + 1$
    $\mathcal{R}_Y \leftarrow \mathcal{R}_Y \cup b$
    Ajouter la contrainte $(*)$
    si $iter < iter\_lim$ **alors**
      Résoudre $P_{\overline{s}_X}$
      si $\sum_{i \in S} x_i < \overline{s}$ **alors**
        $\overline{s}_X \leftarrow \min(\overline{s}_X - 1, \sum_{i=1}^{n} x_i)$
        $iter \leftarrow 1$
      sinon
        $iter \leftarrow iter + 1$
    sinon
      $\overline{s}_X \leftarrow \overline{s}_x - 1$
      $iter \leftarrow 1$
  jusqu'à $\overline{s}_X < n * mincov$
retourner $\mathcal{R}_Y$: ensemble de règles

Sommaire

Comment classifier une donnée/transaction ?

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

Objectif

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

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

Règles nulles

Ajout des règles :

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

Règles nulles

Ajout des règles :

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

Données

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

Variables

- r_l : rang de la règle l $\quad \forall l \in \{1, .., L\}$
- r_* : rang de la plus haute règle nulle

Paramètre

- $R_{rank} \leq \frac{1}{L}$

Objectif

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

- $\sum_{i=1}^{n} \sum_{l=1}^{L} p_{il} u_{il}$: 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 satisfaisant t_i

Contraintes

$$\sum_{i=1}^{L} u_{il} = 1 \qquad \forall l \in \{1, .., L\}$$

$$g_i \geq v_{il} r_l \qquad \forall i \in \{1, .., n\}\ \forall l \in \{1, .., L\}$$

$$g_i \leq v_{il} r_l + L(1 - u_{il}) \qquad \forall i \in \{1, .., n\}\ \forall l \in \{1, .., L\}$$

$$u_{il} \in \{0, 1\} \qquad \forall i \in \{1, .., L\}\ \forall l \in \{1, .., L\}$$

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

$\sum_{k=1}^{L} s_{lk} = 1 \quad \forall l \in \{1, .., L\}$

$\sum_{l=1}^{L} s_{lk} = 1 \quad \forall k \in \{1, .., L\}$

$r_l = \sum_{k=1}^{L} k \, s_{lk} \quad \forall l \in \{1, .., L\}$

$r_l \in \{1, .., L\}$

$s_{lk} \in \{0, 1\}$

Fixation de r_*

Variables

- r_A : rang de $\emptyset \Rightarrow -1$
- r_B : 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

$$r_A = r_{L-1}$$
$$r_B = r_L$$
$$r_* \geq r_A$$
$$r_* \geq r_B$$
$$r_* - r_A \leq (L-1)\alpha$$
$$r_* - r_B \leq (L-1)\beta$$
$$r_A - r_* \leq (L-1)\alpha$$
$$r_B - r_* \leq (L-1)\beta$$
$$\alpha + \beta = 1$$
$$\alpha \in \{0, 1\}$$
$$\beta \in [0, 1]$$

Renforcement de formulation

Contraintes supplémentaires

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

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

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

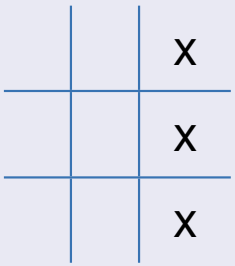
Sommaire

Temps de calcul

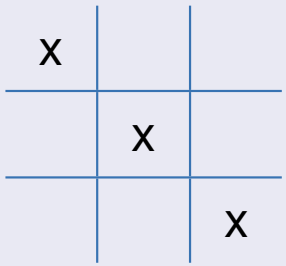
Données	n	d	#Règles	Génération (sec)	Classement (sec)
B.Cancer	683	27	198.3 ± 16.2	616.3 ± 57.8	12959.3 ± 1341.9
CarEval	1728	21	58.0	706.3 ± 177.3	7335.3 ± 2083.7
Crime1	426	41	100.7 ± 15.3	496.0 ± 88.6	12364.0 ± 7100.6
Crime2	436	16	27.3 ± 2.9	59.3 ± 30.4	2546.0 ± 3450.6
Haberman	306	10	15.3 ± 0.6	14.7 ± 4.0	6.3 ± 2.3
Mammo	830	25	58.3 ± 1.2	670.7 ± 34.5	3753.3 ± 3229.5
MONK2	432	17	45.3 ± 4.0	124.0 ± 11.5	5314.3 ± 2873.9
SPECT	267	22	145.3 ± 7.2	71.7 ± 9.1	8862.0 ± 2292.2
TicTacToe	958	27	53.3 ± 3.1	1241.3 ± 38.1	4031.3 ± 3233.0
Titanic	2201	8	24.0 ± 1.0	92.0 ± 15.1	1491.0 ± 1088.0

Résultat du jeu de morpion

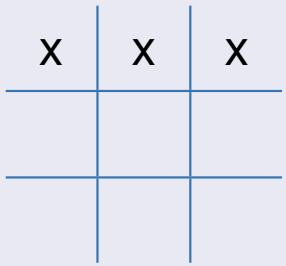
9 règles



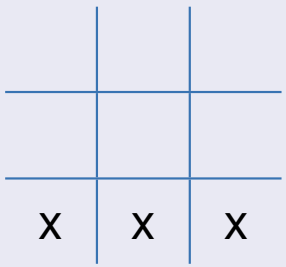
1 - x a gagné



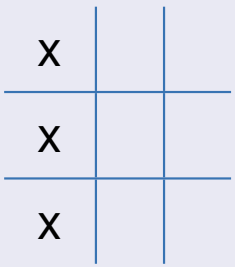
2 - x a gagné



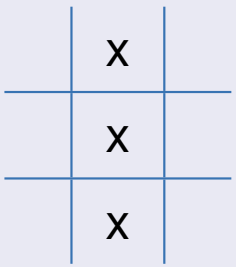
3 - x a gagné



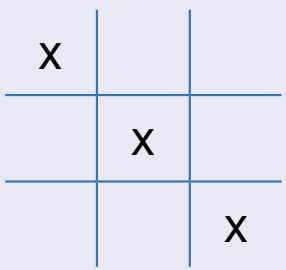
4 - x a gagné



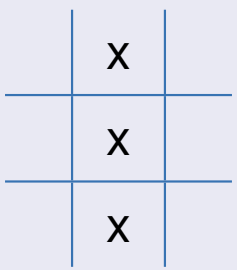
5 - x a gagné



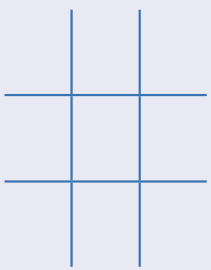
6 - x a gagné



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

Sommaire

Informations générales

Groupe

- seul ou en binôme

Langage

- libre (Julia conseillé)

Calendrier

- 17/01 : 2h15 de TP
- 21/02 : 2h15 de TP (présentation de l'avancement)
- 04/03 : date limite de rendu

Données

306 patients opérés du cancer du sein dont on connaît :

- l'âge
- l'année d'opération
- le nombre de nodules
- s'il a survécu durant les 5 années suivant l'opération (classe du patient)

Objectif

- Utiliser la méthode ORC sur 204 patients pour obtenir un classifieur
- Évaluer ses performances sur les 102 patients restants

caractéristiques			classe
Âge	**Année**	**Nodules**	**A survécu**
30	64	1	1
30	62	3	1
30	65	0	1
⋮	⋮	⋮	⋮
83	58	2	2

Étapes

	caractéristiques			classe
	Âge	**Année**	**Nodules**	**A survécu**
Données initiales 306	30	64	1	1
	30	62	3	1
	30	65	0	1
	⋮	⋮	⋮	⋮
	83	58	2	2

Étapes

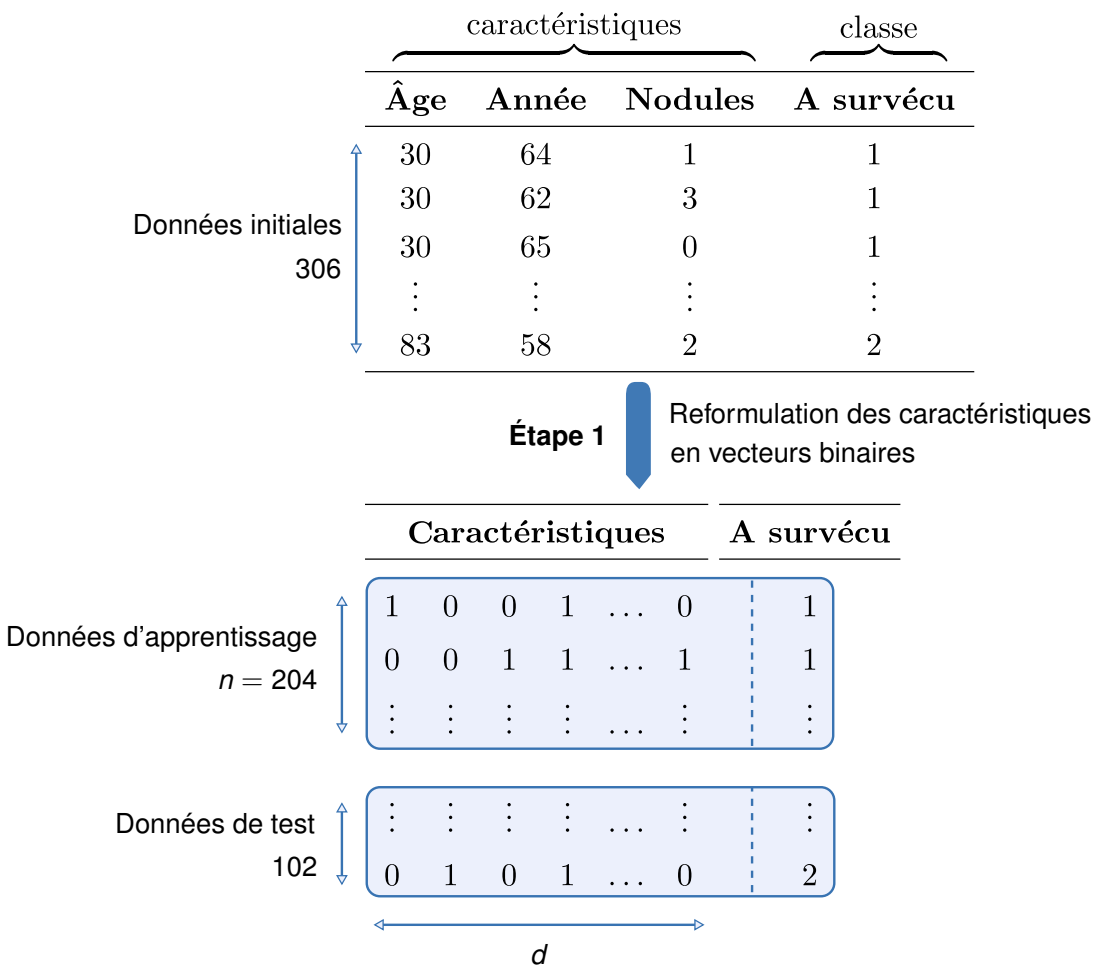
	caractéristiques			classe
	Âge	**Année**	**Nodules**	**A survécu**
Données initiales 306	30	64	1	1
	30	62	3	1
	30	65	0	1
	⋮	⋮	⋮	⋮
	83	58	2	2

Étape 1 Reformulation des caractéristiques en vecteurs binaires

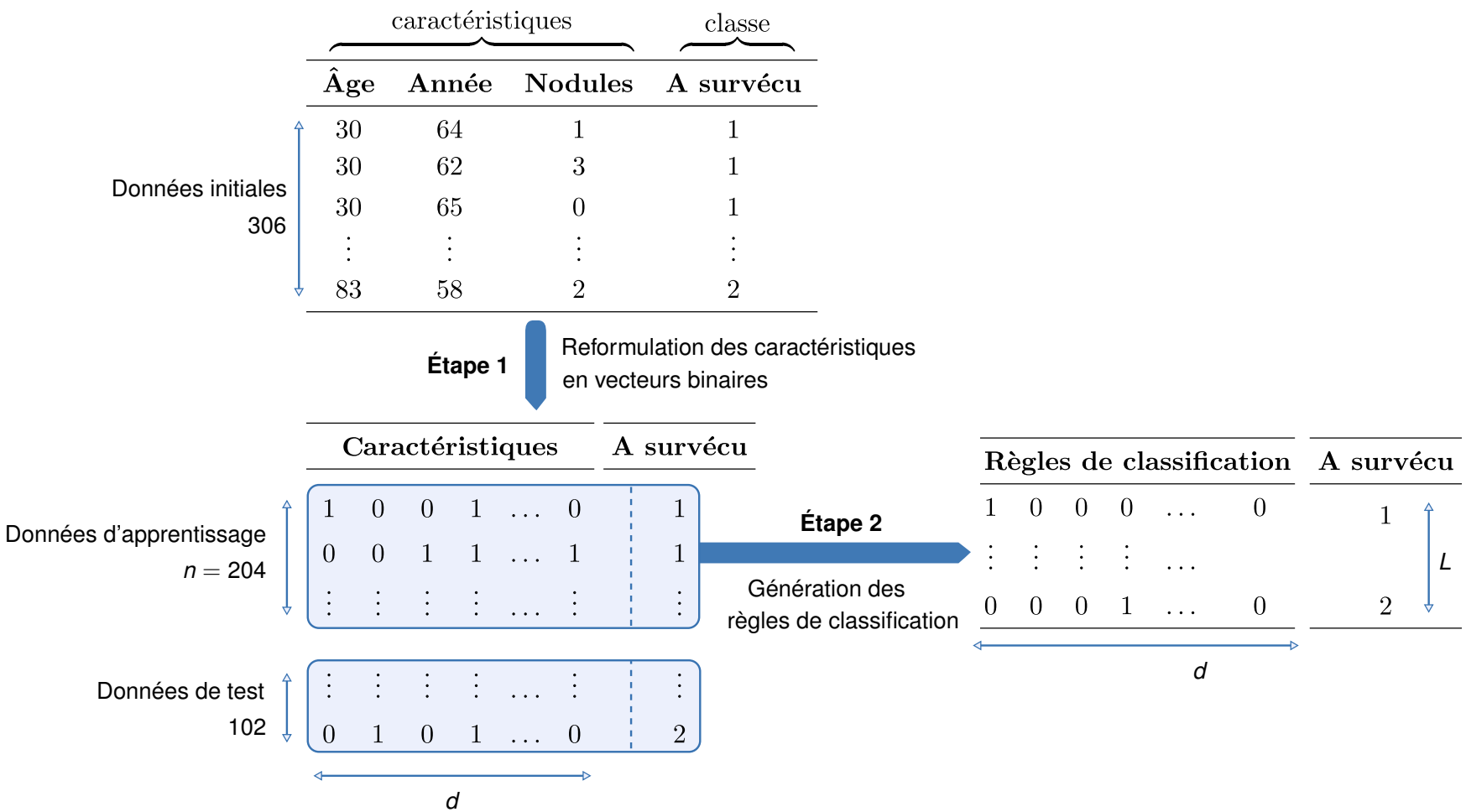
Caractéristiques						**A survécu**
1	0	0	1	…	0	1
0	0	1	1	…	1	1
⋮	⋮	⋮	⋮	…	⋮	⋮
⋮	⋮	⋮	⋮	…	⋮	⋮
0	1	0	1	…	0	2

d

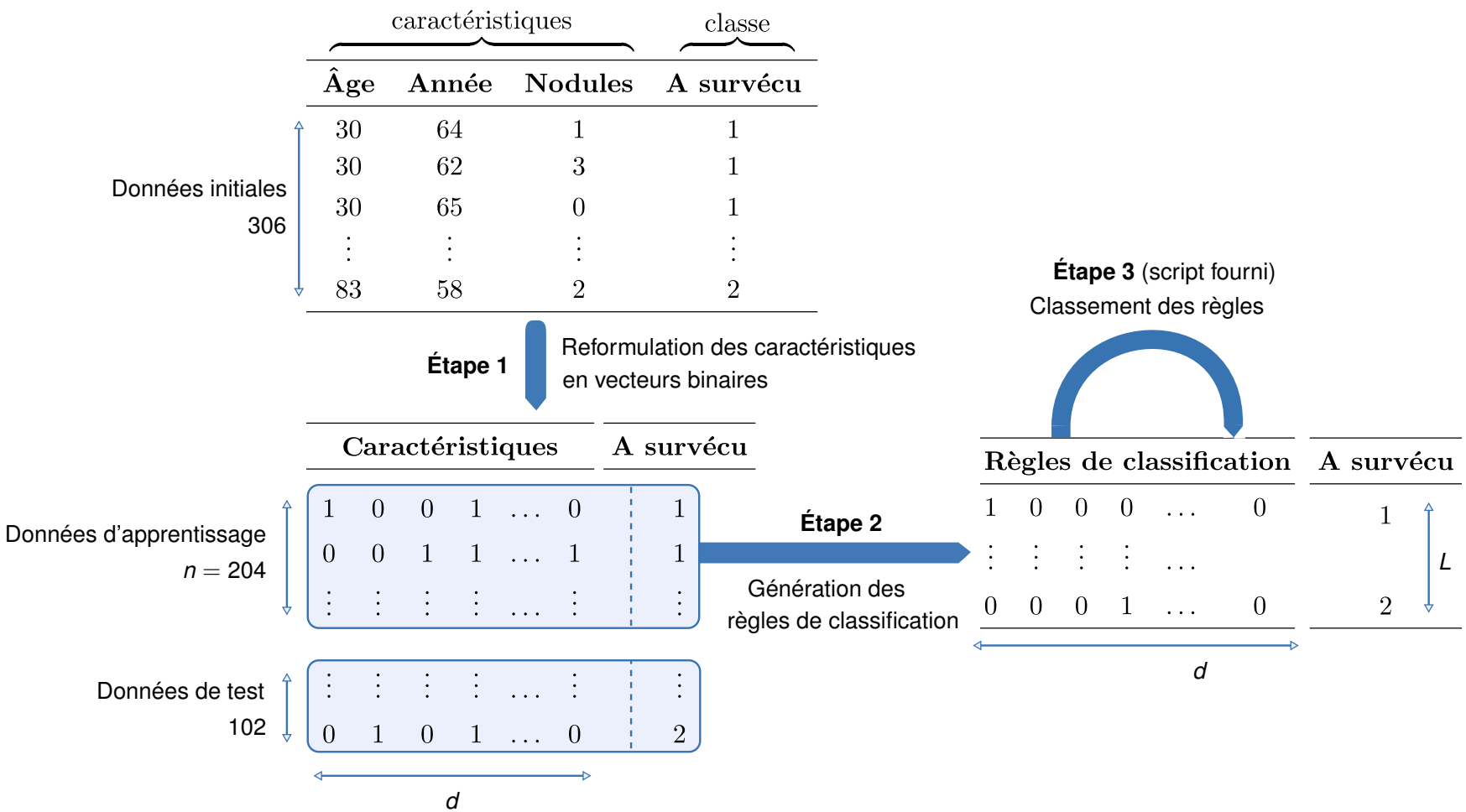
Étapes


caractéristiques
classe
Âge
Année
Nodules
A survécu
30 64 1 1
30 62 3 1
30 65 0 1
83 58 2 2
Données initiales
306
Étape 1
Reformulation des caractéristiques
en vecteurs binaires
Caractéristiques
A survécu
Données d'apprentissage
n = 204
Données de test
102
d

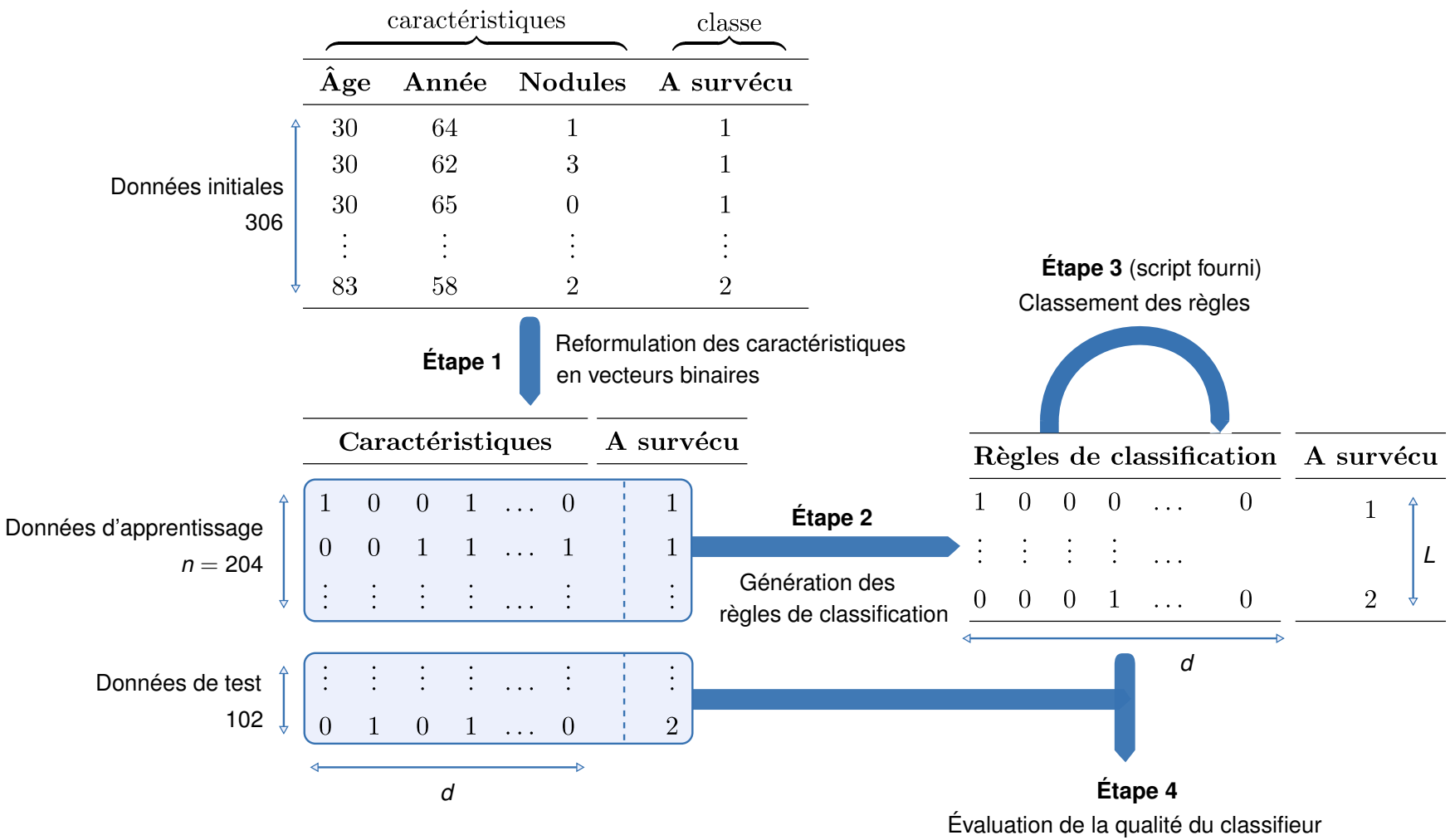
Étapes


caractéristiques
classe
Âge Année Nodules A survécu
30 64 1 1
30 62 3 1
30 65 0 1
83 58 2 2
Données initiales
306
Étape 1
Reformulation des caractéristiques
en vecteurs binaires
Caractéristiques
A survécu
Données d'apprentissage
n = 204
Données de test
102
d
Étape 2
Génération des
règles de classification
Règles de classification
A survécu
L
d

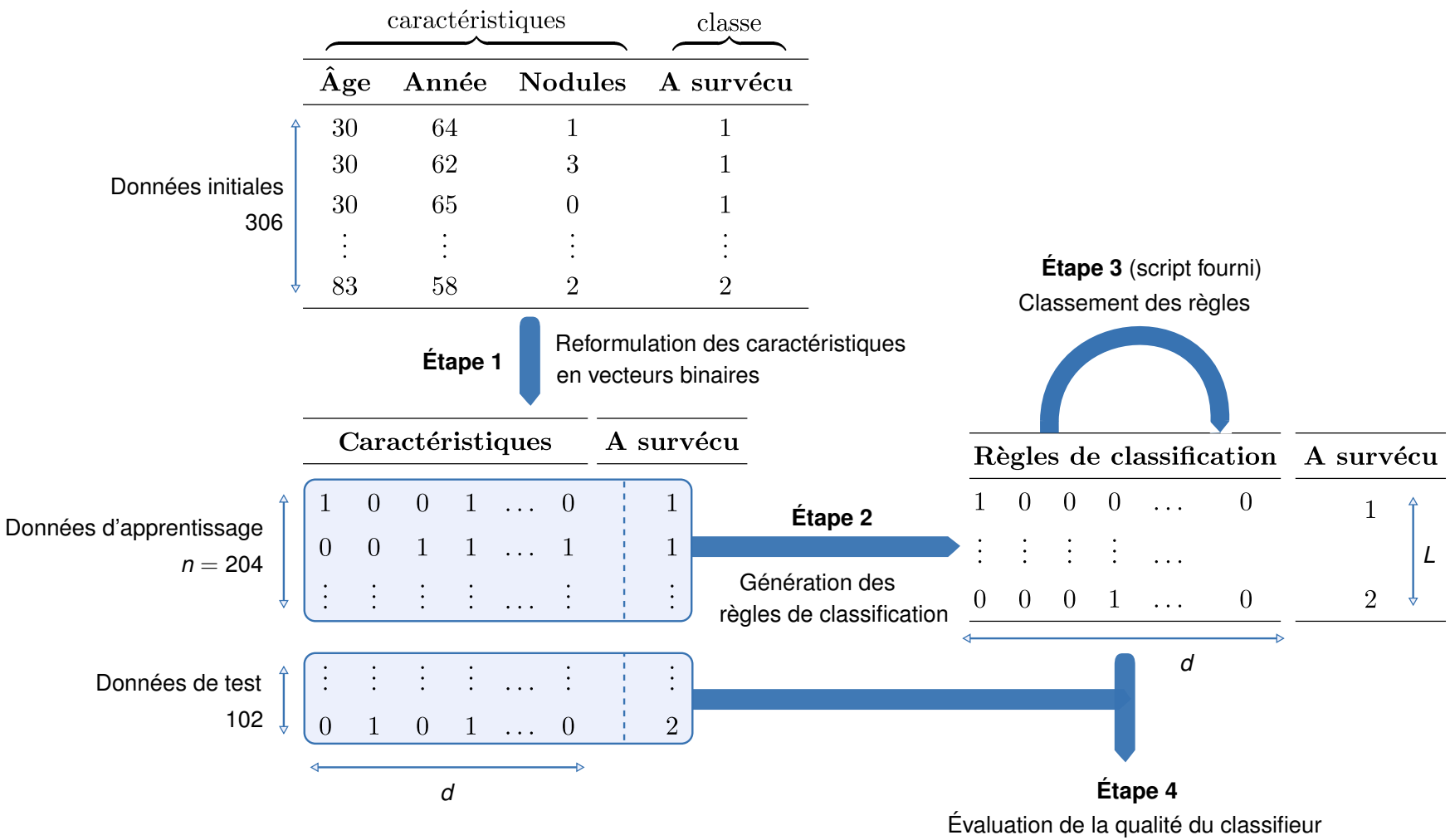
Étapes


caractéristiques
classe
Âge Année Nodules A survécu
30 64 1 1
30 62 3 1
30 65 0 1
83 58 2 2
Données initiales
306
Étape 1
Reformulation des caractéristiques
en vecteurs binaires
Caractéristiques
A survécu
Données d'apprentissage
n = 204
Données de test
102
d
Étape 2
Génération des
règles de classification
Étape 3 (script fourni)
Classement des règles
Règles de classification
A survécu
L
d

Étapes


caractéristiques
classe
Âge Année Nodules A survécu
Données initiales
306
Étape 1
Reformulation des caractéristiques
en vecteurs binaires
Caractéristiques A survécu
Données d'apprentissage
$n = 204$
Données de test
102
d
Étape 2
Génération des
règles de classification
Étape 3 (script fourni)
Classement des règles
Règles de classification A survécu
L
d
Étape 4
Évaluation de la qualité du classifieur

Étapes



Étape 5 - Questions d'ouverture

Structure de votre projet

```
NOM1-NOM2
├─ 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

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
v = 1:n # vecteur de 1 à n
```

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(solver = CplexSolver())
```

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)
```

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

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

$\sum_{i=1}^{n} y_i \geq 3$
```
@constraint(m, 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

```
solve(m)
```

Obtenir la valeur d'une variable

```
vx1 = getvalue(x1)
vx1Int = round(Int, getvalue(x1))
```

Modifier le second membre d'une contrainte

```
@constraint(m, nomDeLaContrainte, x1 + x2 == 1)
JuMP.setRHS(nomDeLaContrainte, 2)
```

Masquer les sorties de CPLEX

```
m = Model(solver=CplexSolver(CPX_PARAM_SCRIND=0))
```

Problème de sac à dos

Fichier donnees.dat

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

Éxecuter ce fichier à l'ENSTA

1. Ouvrir une console : Alt + F2, puis entrer "xterm"
2. Fixer les chemins : usediam ro
3. Éxecuter le programme : julia knapsack.jl

Fichier knapsack.jl

```
using JuMP
using CPLEX

include("donnees.dat")

m = Model(solver = CplexSolver())

@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))

solve(m)
```