Introduction au machine learning

- **caractéristiques** : informations connues (ex : couleur, taux d'alcool, ...)
- **classe** : information qu'on souhaite prédire (ex : bière ou alcool ?)
- **données d'apprentissage** : données utilisées pour apprendre
- **données de test** : données utilisées pour évaluer les performances

1 Génération de règles

- d : nombre de caractéristiques
- $t \in \{0,1\}^d$: transaction représentant une donnée
- n : nombre de transactions dans les données d'apprentissage
- $b \in \{0,1\}^d$: règle
- $b \leq t$: b s'applique à t
- y : classe (d'une transaction ou d'une règle)
- S : ensemble des transactions de classe y
- $x_i = \begin{cases} 1 & \text{si la règle } b \text{ s'applique à } t_i \\ 0 & \text{sinon} \end{cases}$
- couverture : $c = \frac{1}{n} \sum_{i=1}^{n} x_i$
- support : $s = \frac{1}{n} \sum_{i \in S} x_i$
- R_{genX}, R_{genB} : coefficients de l'objectif
- c_{max} : borne supérieure sur la couverture
- $mincov$: valeur minimale de c_{max}
- $P(y, c_{max})$: problème de génération d'une règle pour la classe y dont la couverture est $\leq c_{max}$
- $iter\_lim$: nombre maximum de règles générées pour une valeur de c_{max} donnée
- $\overline{s}$: valeur de l'objectif du dernier problème $P(y, c_{max})$ résolu
- $\mathcal{R}_Y$: ensemble des règles générées

2 Classement des règles générées

- L : nombre de règles générées
- $u_{il} = \begin{cases} 1 & \text{si la règle } l \text{ est la plus haute s'appliquant à } t_i \\ 0 & \text{sinon} \end{cases}$
- $p_{il} = \begin{cases} 0 & \text{si la règle } l \text{ ne s'applique pas à } t_i \\ 1 & \text{si la règle } l \text{ classifie correctement } t_i \\ -1 & \text{si la règle } l \text{ ne classifie pas correctement } t_i \end{cases}$
- $v_{il} = |p_{il}|$
- r_l : rang de la règle l $\forall l \in \{1, .., L\}$
- r_* : rang de la plus haute règle nulle
- R_{rank} : coefficient de l'objectif
- g_i : rang de la plus haute règle s'appliquant à t_i
- s_{lk} : la règle l a le rang k
- r_A : rang de la règle $\emptyset \Rightarrow -1$
- r_B : rang de la règle $\emptyset \Rightarrow 1$
- α : r_* est égal à r_b
- β : r_* est égal à r_a

3 Julia

3.1 Pour utiliser à l'ENSTA

- `# A faire à chaque fois`
  `usediam ro`
- `# A faire uniquement la première fois`
  `julia`
  `using Pkg`
  `Pkg.add("JuMP")`
  `Pkg.add("CPLEX")`
  `Pkg.add("CSV")`
  `Pkg.add("DataFrames")`

3.2 Déclaration

- ```
  # Entier
  n = 10
  ```
- ```
  # Chaîne de caractères
  b = "Hello world"
  ```
- ```
  # Vecteur
  v = [1 2 3 4]
  ```
- ```
  # Matrice 2x2
  m = [1 2; 3 4]
  ```
- ```
  # Vecteur de 1 à n
  v = 1:n
  ```

3.3 Structures de contrôle

- ```
  # Conditionnelle
  if v[1] == 1
  |   # Contenu du if
  else
  |   # Contenu du else
   end
  ```
- ```
  # Boucle for
  for i in 1:10
  |   print(i)
  end
  ```
- ```
  # Boucle while
  while v[1] == 1
  |   # Contenu de la boucle
  end
  ```

3.4 DataFrames

- ```
  # 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)
  ```

3.5 Optimisation

3.5.1 Modèle

- ```
  # Déclaration d'un problème d'optimisation avec CPLEX
  using JuMP
  using CPLEX
  m = Model(with_optimizer(CPLEX.Optimizer))
  ```

3.5.2 Variables

- ```
  # Variable continue
  @variable(m, 0 <= x1 <= 1)
  ```
- ```
  # Variable binaire
  @variable(m, x2, Bin)
  ```
- ```
  # Tableau de taille n*4
  @variable(m, 0 <= t[i in 1:n, j in 1:4] <= 1)
  ```

3.5.3 Contraintes

- # $x_1 + x_2 = 1$
  ```
  @constraint(m, x1 + x2 == 1)
  ```
- # $t_{ij} + x_1 \geq 1 \ \forall i \in \{1, .., n\} \ \forall j \in \{1, .., 4\}$
  ```
  @constraint(m, [i = 1:n, j = 1:4], t[i, j] + x1 >= 1)
  ```
- # $\sum_{i=1}^{n} y_i \geq 3$
  ```
  @constraint(m, sum(y[i] >= 3 for i in 1:n))
  ```

3.5.4 Objectif et résolution

- ```
  # Objectif
  @objective(m, Max, sum(y[i] for i = 1:n if v[i] == 2))
  ```
- ```
  # Résoudre le problème
  optimize!(m)
  ```
- ```
  # Valeur entière d'une variable
  vx1Int = trunc.(Int, JuMP.value(x1))
  ```

3.5.5 Autre

- ```
  # Masquer les sorties de CPLEX
  set_parameter(m, "CPX_PARAM_SCRIND", 0)
  ```
- ```
  # Limiter le temps d'exécution de CPLEX à 30 secondes
  set_parameter(m, "CPX_PARAM_TILIM", 30)
  ```