STT760 : MATHEMATIQUE POUR L'INTELLIGENCE ARTIFICIELLE

Yves Lolelo
Noudéhouénou Houessou
Laetitia Meuleghe Kenmegne
Oussama Khaloui
Gbogou Henri-Michel

2023-2024

STT760 MATHEMATIQUE POUR l'INTELLIGENCE ARTIFICIELLE

2023-10-23

1 - Prépartion des données

```
# install packages ("bnlearn")
library("bnlearn")
data(marks)
names(marks)
```

```
## [1] "MECH" "VECT" "ALG"  "ANL"  "STAT"
```

```
summary(marks)
```

```
##       MECH            VECT            ALG             ANL
##  Min.   : 0.00   Min.   : 9.00   Min.   :15.00   Min.   : 9.00
##  1st Qu.:30.00   1st Qu.:42.00   1st Qu.:45.00   1st Qu.:35.75
##  Median :41.50   Median :51.00   Median :50.00   Median :49.00
##  Mean   :38.95   Mean   :50.59   Mean   :50.60   Mean   :46.68
##  3rd Qu.:49.25   3rd Qu.:60.00   3rd Qu.:57.25   3rd Qu.:57.00
##  Max.   :77.00   Max.   :82.00   Max.   :80.00   Max.   :70.00
##       STAT
##  Min.   : 9.00
##  1st Qu.:31.00
##  Median :40.00
##  Mean   :42.31
##  3rd Qu.:51.50
##  Max.   :81.00
```

```
notes_reussite = (marks >=45)*1
notes_reussite[notes_reussite==1] = "R"
notes_reussite[notes_reussite==0] = "E"
```

Dans la matrice **notes_reussite**, un **R** indique une réussite et un **E** indique un échec.

2- Fonctions pour paramétriser $\mathbb{P}_{Xi}$

$X_i = (X_{i1}, ..., X_{i5})$ est un vecteur aléatoire tel que X_{i1} (respectivement $(X_{i2}..., X_{i5})$) vaut 1 si l'étudiant(e) i a réussi mécanique (resp. algèbre vectorielle, algèbre, analyse et statistique)

a) Elicitons les fonctions f_1, f_2, f_3, f_4 qui permettent de paramétriser $\mathbb{P}_{Xi}$

```
f1 <- function(x,p1){
  e=0
  if((x==0) || (x==1))
  {
    e = (p1^x) * (1-p1)^(1-x)
```

```
  }
  return(e)
}
```

```
f2 <- function(x,y,p2,p3){
  e=0
  if((x==0) || (x==1) & ((y==0) || (y==1)))
  {
    e = ((p2^x * (1-p2)^(1-x))^y) * ((p3^x * (1-p3)^(1-x))^(1-y))
  }
  return(e)
}
```

```
f3 <- function(x,y,z,p4,p5,p6,p7) {
   e=0
   f=0
   g=0
   h=0
  if((x==0) || (x==1) & ((y==0) || (y==1)) & ((z==0) || (z==1)))
  {
   e = p4^x * (1-p4)^(1-x)
   f = p5^x * (1-p5)^(1-x)
   g = p6^x * (1-p6)^(1-x)
   h = p7^x * (1-p7)^(1-x)
   t = (e^(y*z) * f^(z*(1-y))* g^(y*(1-z))* h^((1-y)*(1-z)))
  }
  return (t)
}
```

```
f4 <- function(x,y,p8,p9){
  e=0
  if((x==0) || (x==1) & ((y==0) || (y==1)))
  {
    e = ((p8^x * (1-p8)^(1-x))^y) * ((p9^x * (1-p9)^(1-x))^(1-y))
  }
  return (e)
}
```

```
f5 <- function(x,y,z,p10,p11,p12,p13) {
   e=0
   f=0
   g=0
   h=0
  if((x==0) || (x==1) & ((y==0) || (y==1)) & ((z==0) || (z==1)))
  {
   e = p10^x * (1-p4)^(1-x)
   f = p11^x * (1-p11)^(1-x)
   g = p12^x * (1-p12)^(1-x)
   h = p13^x * (1-p13)^(1-x)
   t = (e^(y*z)*f^(z*(1-y))*g^(y*(1-z))*h^((1-y)*(1-z)))
  }
  return (t)
}
```

b) La fonction de vraisemblance se présente comme suit :

```
L_p <- function(x,y,z,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13){

  return(f1(x,p1)*f2(x,y,p2,p3)*f3(x,y,z,p4,p5,p6,p7)*f4(x,y,p8,p9)*
           f5(x,y,z,p10,p11,p12,p13))
}
```

3 - Ajustement de paramètres par maximum d evraisemblance

Formellement, étant donné un échantillon d'observations indépendantes et de même distribution, on a :

$$p^* = argmax \prod_{i=1}^{n} L(x_i, p)$$

- De la question précédente, la fonction de vraisemblance se présente comme suit :

```
L_p <- function(x,y,z,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13){

  return(f1(x,p1)*f2(x,y,p2,p3)*f3(x,y,z,p4,p5,p6,p7)*f4(x,y,p8,p9)*
           f5(x,y,z,p10,p11,p12,p13))
}
```

- Les différentes valeurs de p^* obtenues analytiquement se présentent comme suit :

```
Phat_1 <- function(X5) {return(sum(X5)/length(X5))}
```

```
Phat_2 <- function(X4,X5) {return(sum(X4*X5)/sum(X5))}
```

```
Phat_3 <- function(X4,X5) { return(sum((1-X5)*X4)/sum(1-X5))}
```

```
Phat_4 <- function(X3, X4, X5) { return(sum(X3*X4*X5)/sum(X4*X5))}
```

```
Phat_5 <- function(X3, X4, X5) {return(sum(X3*X4*(1-X5))/sum(X4*(1-X5)))}
```

```
Phat_6 <- function(X3, X4, X5) { return(sum(X3*X5*(1-X4))/sum((1-X4)*X5))}
```

```
Phat_7 <- function(X3, X4, X5) {return(sum(X3*(1-X4)*(1-X5))/sum((1-X4)*(1-X5)))}
```

```
Phat_8 <- function(X2,X3) {return(sum(X2*X3)/sum(X3))}
```

```
Phat_9 <- function(X2,X3) {return(sum(X2*(1-X3))/sum(1-X3))}
```

```
Phat_10 <- function(X1, X2, X3) {return(sum(X1*X2*X3)/sum(X2*X3))}
```

```
Phat_11 <- function(X1, X2, X3) {return(sum(X1*X2*(1-X3))/sum(X2*(1-X3)))}
```

```
Phat_12 <- function(X1, X2, X3) {return(sum(X1*(1-X2)*X3)/sum(X3*(1-X2)))}
```

```
Phat_13 <- function(X1, X2, X3) {return(sum(X1*(1-X2)*(1-X3))/sum((1-X2)*(1-X3)))}
```

- Calculons la valeur de p^*

```
notes_reussite = (marks >=45)*1
X1 <- notes_reussite[,1]
X2 <- notes_reussite[,2]
X3 <- notes_reussite[,3]
X4 <- notes_reussite[,4]
X5 <- notes_reussite[,5]
```

```
Phat <- c(Phat_1(X5), Phat_2(X4, X5), Phat_3(X4, X5), Phat_4(X3, X4, X5),
          Phat_5(X3,X4,X5), Phat_6(X3,X4,X5), Phat_7(X3,X4,X5), Phat_8(X2, X3),
          Phat_9(X2, X3), Phat_10(X1,X2,X3), Phat_11(X1,X2,X3), Phat_12(X1,X2,X3), Phat_13(X1,X2,X3))

Phat
```

```
##  [1] 0.3863636 0.8529412 0.5555556 0.9655172 0.8333333 0.8000000 0.4166667
##  [8] 0.7761194 0.2857143 0.5576923 0.3333333 0.3333333 0.1333333
```

4- Comparons la solution trouvée avec le résultat de la fonction *bn.fit*

La fonction *bn.fit* est contenue dans la librairie **bnlearn** qui représente un réseau bayésien par l'entremise d'un objet de type *bn*. Pour ce faire :

- (a) Construisons un graphe $\mathcal{G}$ sans arête à partir d'une liste de noeuds

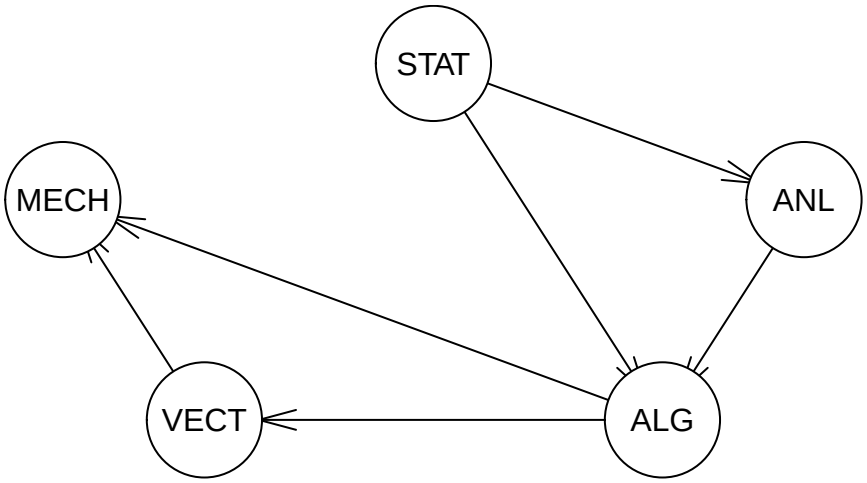
```
graphe_notes = empty.graph(names(marks))
```

- (b) Modifions un graphe $\mathcal{G}$ en y ajoutant des arêtes

```
arcs(graphe_notes) = matrix(
c("VECT", "MECH", "ALG", "MECH", "ALG", "VECT",
  "ANL", "ALG", "STAT", "ALG","STAT", "ANL"),
   ncol = 2, byrow = TRUE, dimnames = list(c(), c("from", "to")))
graphe_notes
```

```
##
##   Random/Generated Bayesian network
##
##   model:
##    [STAT][ANL|STAT][ALG|ANL:STAT][VECT|ALG][MECH|VECT:ALG]
##   nodes:                                 5
##   arcs:                                  6
##     undirected arcs:                     0
##     directed arcs:                       6
##   average markov blanket size:           2.40
##   average neighbourhood size:            2.40
##   average branching factor:              1.20
##
##   generation algorithm:                  Empty
```

```
plot(graphe_notes)
```



- (c) Spécifions les paires d'arêtes en élicitant la matrice d'adjacence associée au graphe $\mathcal{G}$ comme c'est le cas pour les objets *GraphNEL*

```
matrice_ad = matrix(c(0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
                      0, 0, 1, 0, 0, 0, 0, 0), nrow = 5,
                    dimnames = list(names(marks), names(marks)))
matrice_ad
```

```
##      MECH VECT ALG ANL STAT
## MECH    0    0   0   0    0
## VECT    1    0   0   0    0
## ALG     1    1   0   0    0
## ANL     0    0   1   0    0
## STAT    0    0   1   1    0
```

Ainsi, la strucutre du réseau est créée.

```
graphe2_notes = empty.graph(names(marks))
amat(graphe2_notes) = matrice_ad
all.equal(graphe_notes, graphe2_notes)
```

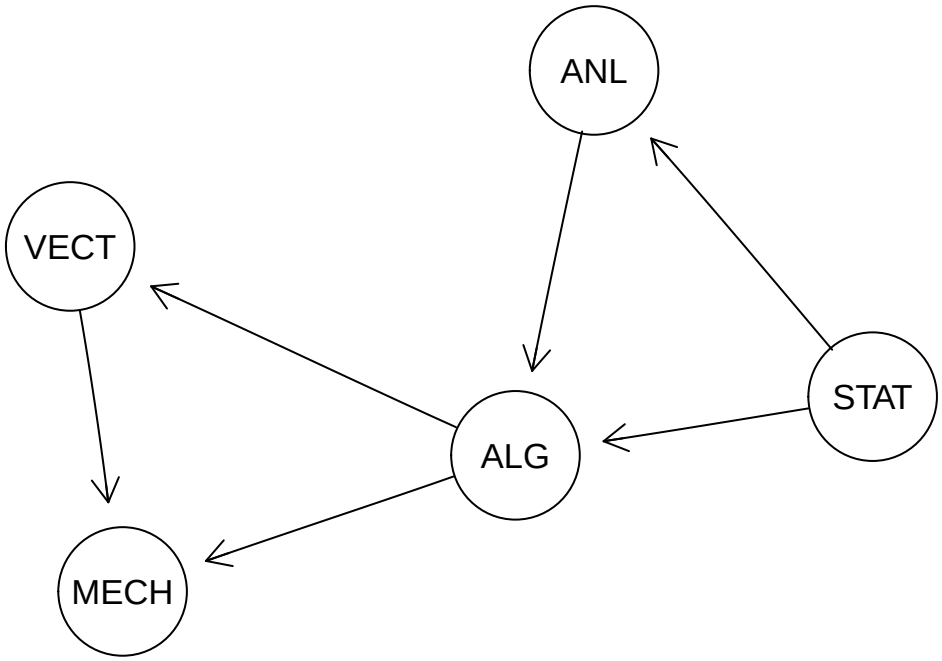
```
## [1] TRUE
```

- (d) Visualisons la structure de réseau Bayésien

```
v_stb = list(arcs = vstructs(graphe_notes, arcs = TRUE), lwd = 4, col = "black")
graphviz.plot(graphe_notes, highlight = v_stb, layout = "fdp", main = "Interdépendances des sujets")
```

```
## Le chargement a nécessité le package : Rgraphviz
```

```
## Warning in graphviz.backend(nodes = nodes, arcs = arcs, highlight = highlight,
## : no arc to apply the 'lwd' setting to, ignoring.
```



- (e) Vérifions les paramètres du modèle

```
notes_reussite = (marks >=45)*1
bn.fit(graphe_notes, data = as.data.frame(notes_reussite))
```

```
##
##   Bayesian network parameters
##
##   Parameters of node MECH (Gaussian distribution)
##
## Conditional density: MECH | VECT + ALG
## Coefficients:
## (Intercept)         VECT          ALG
##   0.1282463    0.2178046    0.2101740
## Standard deviation of the residuals: 0.4756269
##
##   Parameters of node VECT (Gaussian distribution)
##
## Conditional density: VECT | ALG
## Coefficients:
## (Intercept)          ALG
##   0.2857143    0.4904051
## Standard deviation of the residuals: 0.4303528
##
##   Parameters of node ALG (Gaussian distribution)
##
## Conditional density: ALG | ANL + STAT
## Coefficients:
## (Intercept)          ANL         STAT
##   0.4504797    0.3558032    0.1872176
```

```
## Standard deviation of the residuals: 0.3752541
##
##   Parameters of node ANL (Gaussian distribution)
##
## Conditional density: ANL | STAT
## Coefficients:
## (Intercept)         STAT
##   0.5555556    0.2973856
## Standard deviation of the residuals: 0.4523587
##
##   Parameters of node STAT (Gaussian distribution)
##
## Conditional density: STAT
## Coefficients:
## (Intercept)
##   0.3863636
## Standard deviation of the residuals: 0.4897059
```