exam_2019_mysol

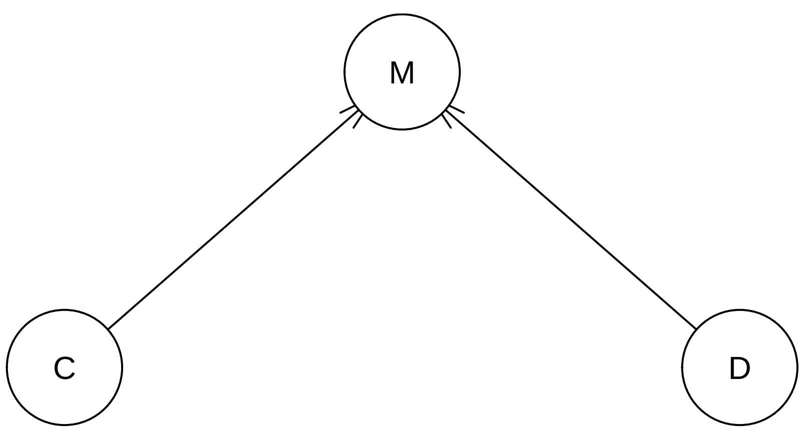
David Björelind

10/22/2020

1. Graphical Models

a)

```
#C: Car
#D: Door of choice
#M: Monty's choice

# Making the network model
graph = model2network("[D][C][M|C:D]")
plot(graph)
```



```r
# Making the Conditional Probability Tables
cptC = matrix(c(1/3, 1/3, 1/3), ncol = 3, dimnames = list(NULL, c("Car1", "Car2", "Car3")))

cptD = matrix(c(1/3, 1/3, 1/3), ncol = 3, dimnames = list(NULL, c("Choice1", "Choice2", "Choice3")))

cptM = c(0, 0.5, 0.5,
          0, 0, 1,
          0, 1, 0,
          0, 0, 1,
          0.5, 0, 0.5,
          1, 0, 0,
          0, 1, 0,
          1, 0, 0,
          0.5, 0.5, 0)
dim(cptM) = c(3,3,3)
dimnames(cptM) = list("M" = c("Door1", "Door2", "Door3"), "D" =  c("Choice1", "Choice2", "Choice3"), "C

dist = list("D" = cptD, "C"= cptC, "M" = cptM) # Largest one needs to be last and names needs to be the
parameters = custom.fit(graph, dist = list("D" = cptD, "C"= cptC, "M" = cptM))

### EXAKT INFERENCE ###
grain = as.grain(parameters)
structure = compile(grain) # creating junction tree, separators & residuals. Potentials
goal = c("C")
evi = setEvidence(structure, nodes = c(""), states = c(""))
dist = querygrain(evi, nodes = goal)
# Picking door 1, monty 2
evi = setEvidence(structure, nodes = c("D", "M"), states = c("Choice1", "Door2"))
querygrain(evi, nodes = goal)
```

```
## $C
## C
##      Car1      Car2      Car3
## 0.3333333 0.0000000 0.6666667
```

```r
# Picking door 3, monty 2
evi = setEvidence(structure, nodes = c("D", "M"), states = c("Choice3", "Door2"))
querygrain(evi, nodes = goal)
```

```
## $C
## C
##      Car1      Car2      Car3
## 0.6666667 0.0000000 0.3333333
```

```r
evi = setEvidence(structure, nodes = c("D", "M"), states = c("Choice1", "Door3"))
querygrain(evi, nodes = goal)
```

```
## $C
## C
##      Car1      Car2      Car3
## 0.3333333 0.6666667 0.0000000
```

```r
### APPROXIMATE INFERENCE ###

a = cpdist(fitted = parameters, nodes = "C", evidence = TRUE)
table(a)/sum(table(a))
```

```
## a
##   Car1   Car2   Car3
## 0.3426 0.3242 0.3332
```

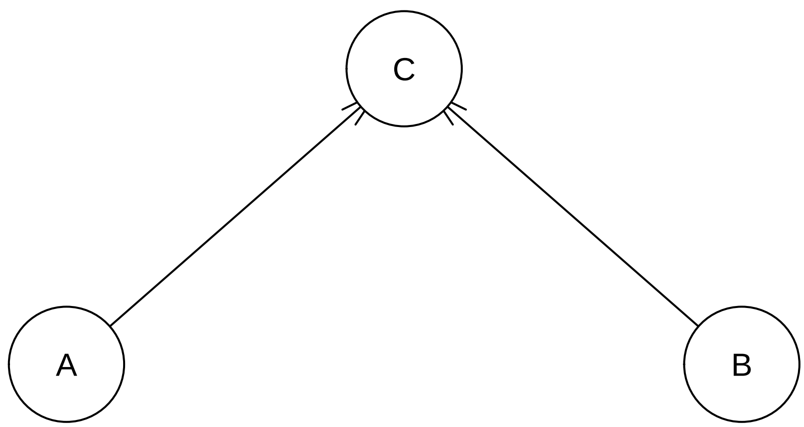
```r
b =  cpdist(fitted = parameters,nodes = "C", evidence = (D=="Choice1" & M=="Door2"))
table(b)/sum(table(b))
```

```
## b
##      Car1      Car2      Car3
## 0.3320849 0.0000000 0.6679151
```

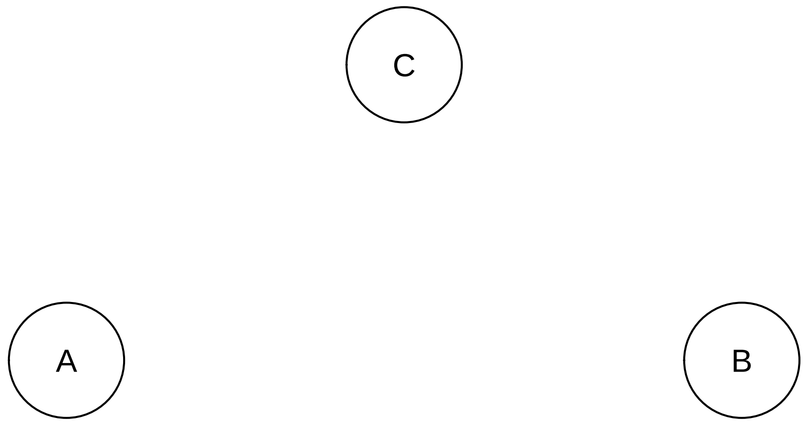
Conclusion: Always switch doors!!

b)

```r
# Making the network model
graph = model2network("[B][A][C|A:B]")
plot(graph)
```



```r
# Making the Conditional Probability Tables
cptA = matrix(c(1/2, 1/2), ncol = 2, dimnames = list(NULL, c("A0", "A1")))

cptB = matrix(c(1/2, 1/2), ncol = 2, dimnames = list(NULL, c("B0", "B1")))

cptC = c(1, 0,
         0, 1,
         0, 1,
         1, 0)
dim(cptC) = c(2,2,2)
dimnames(cptC) = list("C" = c("C0", "C1"), "A" =  c("A0", "A1"), "B" = c("B0", "B1"))

dist = list("A" = cptA, "B"= cptB, "C" = cptC) # Largest one needs to be last and names needs to be the
parameters = custom.fit(graph, dist = list("A" = cptA, "B"= cptB, "C" = cptC))
niter = 1000
# Drawing samples
sample = rbn(parameters, n=niter)

# Learning HC from samples
learn_graph = hc(sample)

# Repeating 10 times
for (i in 1:10){
  sample = rbn(parameters, n=niter)
  learn_graph = hc(sample)
  plot(learn_graph)
}
```



Why does HC fail to recover the true BN structure in most runs?

The algorithm gets stuck on a local optimum? Since HC starts in a random spot (and we're not allowed to use random restarts) it can get stuck. HC is also Score-based.

Edge from A -> C but A is also marginally dependant of C. That is why the algorithm can't find two dependant variables!

2. Hidden Markov Models

```
# Building a Hidden Markov Model
state = rep(1:10) # Actual number of states that the robot can be in (Hidden)
probs = rep(1/10, 10)
symbols = 1:11 # States that we can observe (Not hidden)

# If robot is in sector i:
emissionP = matrix(c(
  0.1, 0.1, 0.1, 0, 0, 0, 0, 0, 0.1, 0.1, 0.5,
  0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0, 0, 0.1, 0.5,
  0.1, 0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0, 0, 0.5,
  0, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0, 0.5,
  0, 0, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0.5,
  0, 0, 0, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0, 0.5,
  0, 0, 0, 0, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0.5,
  0, 0, 0, 0, 0, 0.1, 0.1, 0.1, 0.1, 0.1, 0.5,
  0.1, 0, 0, 0, 0, 0, 0.1, 0.1, 0.1, 0.1, 0.5,
```

```r
  0.1, 0.1, 0, 0, 0, 0, 0, 0.1, 0.1, 0.1, 0.5),
  ncol = 11, byrow = TRUE
)

#transP = matrix(c(
#  0.1, 0.1, 0.1, 0, 0, 0, 0, 0, 0.1, 0.1, 0,
#  0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0, 0, 0.1, 0,
#  0.1, 0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0, 0, 0,
#  0, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0, 0,
#  0, 0, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0,
#  0, 0, 0, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0, 0,
#  0, 0, 0, 0, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0,
#  0, 0, 0, 0, 0, 0.1, 0.1, 0.1, 0.1, 0.1, 0,
#  0.1, 0, 0, 0, 0, 0, 0.1, 0.1, 0.1, 0.1, 0,
#  0.1, 0.1, 0, 0, 0, 0, 0, 0.1, 0.1, 0.1, 0,
#  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
#  ncol = 11, byrow = TRUE
#)
transP = matrix(c(
  0.5, 0.5, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0.5, 0.5, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0.5, 0.5, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0.5, 0.5, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0.5, 0.5, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0.5, 0.5, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0.5, 0.5, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0.5, 0.5, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0.5, 0.5,
  0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0.5),
  ncol = 10, byrow = TRUE
)
# Initializing hidden markov model
robot = initHMM(States = state, Symbols = symbols, startProbs = probs, transProbs = transP, emissionPro

# Defining path
obs = c(1, 11, 11, 11)

# Most probable path, using VITERBI algorithm
posterior(robot, obs)
```

```
##       index
## states   1   2    3     4
##     1  0.2 0.2 0.20 0.175
##     2  0.2 0.2 0.20 0.200
##     3  0.2 0.2 0.20 0.200
##     4  0.0 0.1 0.15 0.175
##     5  0.0 0.0 0.05 0.100
##     6  0.0 0.0 0.00 0.025
##     7  0.0 0.0 0.00 0.000
##     8  0.0 0.0 0.00 0.000
##     9  0.2 0.1 0.05 0.025
##     10 0.2 0.2 0.15 0.100
```

```r
viterbi(robot, obs)
```

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

According to Viterbi, most probable path forthe scenario is for the robot to stay in sector 1 all the time.

3. Reinforcement Learning

```r
# No assignment :(((
```

4. Gaussian Processes

a) Extension of lab

Another chunk