TDDE15- Exam Oct 2023

Fredrik Ramberg

1

```r
library(bnlearn)
library(gRain)
```

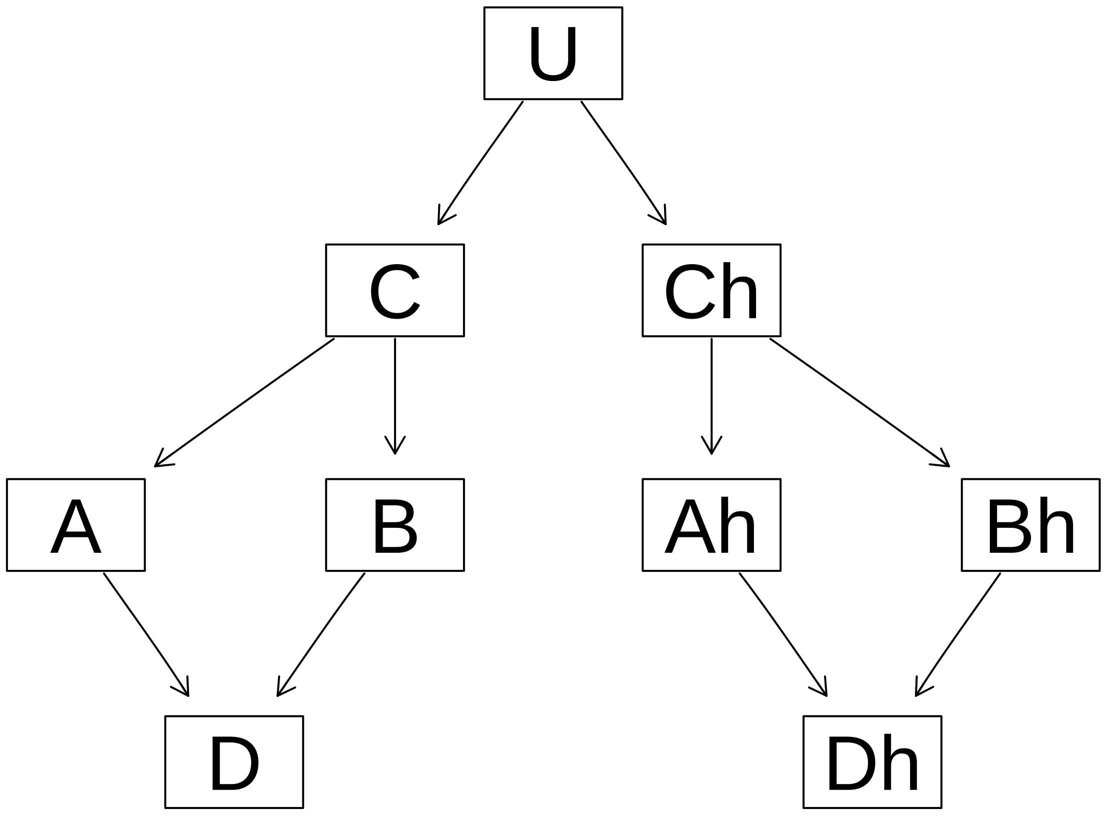
```
## Loading required package: gRbase

##
## Attaching package: 'gRbase'

## The following objects are masked from 'package:bnlearn':
##
##     ancestors, children, nodes, parents
```

```r
dag <- model2network("[U][C|U][A|C][B|C][D|A:B][Ch|U][Ah|Ch][Bh|Ch][Dh|Ah:Bh]")
graphviz.plot(dag)
```

```
## Loading required namespace: Rgraphviz
```



```
cptU <- c(.5,.5)
dim(cptU) <- c(2)
dimnames(cptU) <- list(c("0", "1"))

#### C
## U
cptC <- matrix(c(.9,.1,
                 .1,.9), nrow=2, ncol=2)
dim(cptC) <- c(2,2)
dimnames(cptC) <- list("C" = c("0", "1"), "U" =  c("0", "1"))

cptA <- matrix(c(1,0,
                 .2,.8), nrow=2, ncol=2)
dim(cptA) <- c(2,2)
dimnames(cptA) <- list("A" = c("0", "1"), "C" =  c("0", "1"))

cptB <- matrix(c(1,0,
                 .2,.8), nrow=2, ncol=2)
dim(cptB) <- c(2,2)
dimnames(cptB) <- list("B" = c("0", "1"), "C" =  c("0", "1"))

#### D    #B = 0
## A

#### D    #B = 1
## A

cptD <- matrix(c(.9,.1
                 ,0,1,
                0,1,
                0,1), nrow=2, ncol=4)
dim(cptD) <- c(2,2,2)
dimnames(cptD) <- list("D" = c("0", "1"), "A" =  c("0", "1"), "B" =  c("0", "1"))

cptCh <- matrix(c(.9,.1,
                 .1,.9), nrow=2, ncol=2)
dim(cptCh) <- c(2,2)
dimnames(cptCh) <- list("Ch" = c("0", "1"), "U" =  c("0", "1"))

cptAh <- matrix(c(1,0,
                 .2,.8), nrow=2, ncol=2)
dim(cptAh) <- c(2,2)
dimnames(cptAh) <- list("Ah" = c("0", "1"), "Ch" =  c("0", "1"))

cptBh <- matrix(c(1,0,
                 .2,.8), nrow=2, ncol=2)
dim(cptBh) <- c(2,2)
dimnames(cptBh) <- list("Bh" = c("0", "1"), "Ch" =  c("0", "1"))

#### D    #B = 0
## A

#### D    #B = 1
```

```r
## A

cptDh <- matrix(c(.9,.1
                ,0,1,
                0,1,
                0,1), nrow=2, ncol=4)
dim(cptDh) <- c(2,2,2)
dimnames(cptDh) <- list("Dh" = c("0", "1"), "Ah" =  c("0", "1"), "Bh" =  c("0", "1"))

bn <- custom.fit(dag, list(U=cptU,C=cptC,A=cptA,B=cptB,D=cptD,
                                  Ch=cptCh,Ah=cptAh,Bh=cptBh,Dh=cptDh))
bn <- compile(as.grain(bn))


posterior <- querygrain(setEvidence(bn, nodes = c("D", "Ah"), states = c("1", "0")), nodes = "Dh")
posterior$Dh[2]
```

```
##         1
## 0.3790428
```

2

```r
#install.packages("HMM")
library(HMM)
#states = hidden states
states <- c("1a", "1b", "2a", "2b", "2c", "3a", "3b", "4a", "5a", "5b")

#symbols = observations
symbols <- c(1:5)

emissionProbs <- c(1/3,1/3,  0,  0,1/3,
                   1/3,1/3,  0,  0,1/3,
                   1/3,1/3,1/3,  0,  0,
                   1/3,1/3,1/3,  0,  0,
                   1/3,1/3,1/3,  0,  0,
                     0,1/3,1/3,1/3,  0,
                     0,1/3,1/3,1/3,  0,
                     0,  0,1/3,1/3,1/3,
                   1/3,  0,  0,1/3,1/3,
                   1/3,  0,  0,1/3,1/3)
######## Symbols
# States
emissionProbs <- matrix(emissionProbs, ncol = 5, byrow = TRUE)

transProbs <- 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)
##### Old
# New
transProbs <- matrix(transProbs,nrow = 10, ncol = 10, byrow = TRUE)

hmm <- initHMM(states,symbols, transProbs = transProbs, emissionProbs = emissionProbs)

transProbs
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]  0.5  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0
##  [2,]  0.0  0.5  0.5  0.0  0.0  0.0  0.0  0.0  0.0   0.0
##  [3,]  0.0  0.0  0.5  0.5  0.0  0.0  0.0  0.0  0.0   0.0
##  [4,]  0.0  0.0  0.0  0.5  0.5  0.0  0.0  0.0  0.0   0.0
##  [5,]  0.0  0.0  0.0  0.0  0.5  0.5  0.0  0.0  0.0   0.0
##  [6,]  0.0  0.0  0.0  0.0  0.0  0.5  0.5  0.0  0.0   0.0
##  [7,]  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.5  0.0   0.0
##  [8,]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.5   0.0
##  [9,]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5   0.5
## [10,]  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.5
```

```r
emissionProbs
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]
##  [1,] 0.3333333 0.3333333 0.0000000 0.0000000 0.3333333
##  [2,] 0.3333333 0.3333333 0.0000000 0.0000000 0.3333333
##  [3,] 0.3333333 0.3333333 0.3333333 0.0000000 0.0000000
##  [4,] 0.3333333 0.3333333 0.3333333 0.0000000 0.0000000
##  [5,] 0.3333333 0.3333333 0.3333333 0.0000000 0.0000000
##  [6,] 0.0000000 0.3333333 0.3333333 0.3333333 0.0000000
##  [7,] 0.0000000 0.3333333 0.3333333 0.3333333 0.0000000
##  [8,] 0.0000000 0.0000000 0.3333333 0.3333333 0.3333333
##  [9,] 0.3333333 0.0000000 0.0000000 0.3333333 0.3333333
## [10,] 0.3333333 0.0000000 0.0000000 0.3333333 0.3333333
```

```r
set.seed(12345)
simulation100 <- simHMM(hmm, 100)
simulation100
```

```
## $states
##   [1] "5a" "5a" "5a" "5a" "5b" "1a" "1b" "1b" "1b" "1b" "2a" "2a" "2b" "2b" "2b"
##  [16] "2b" "2b" "2b" "2b" "2c" "3a" "3a" "3b" "4a" "5a" "5b" "5b" "5b" "1a" "1b"
##  [31] "1b" "2a" "2a" "2b" "2b" "2b" "2c" "2c" "2c" "3a" "3b" "3b" "4a" "5a" "5b"
##  [46] "1a" "1b" "2a" "2a" "2b" "2c" "2c" "3a" "3a" "3b" "3b" "4a" "4a" "4a" "4a"
##  [61] "5a" "5b" "5b" "5b" "5b" "1a" "1a" "1b" "1b" "1b" "1b" "1b" "2a" "2a" "2a"
##  [76] "2b" "2c" "2c" "2c" "3a" "3b" "4a" "4a" "4a" "4a" "4a" "5a" "5a" "5a" "5b"
##  [91] "5b" "5b" "1a" "1a" "1a" "1a" "1a" "1a" "1b" "2a"
##
## $observation
##   [1] 1 5 4 5 5 2 5 2 1 1 3 3 2 3 2 2 2 3 1 1 3 3 2 4 4 4 4 1 1 2 1 2 2 3 2 2 2
##  [38] 1 1 4 3 4 5 5 1 2 2 3 2 3 2 3 4 3 2 4 4 3 4 3 4 5 4 4 5 5 2 1 2 5 2 1 2 1
##  [75] 3 2 2 2 2 4 2 3 5 5 4 4 5 4 1 1 4 4 1 1 2 1 1 2 1 3
```

```r
table(simulation100$states, simulation100$observation)
```

```
##
##      1 2 3 4 5
##   1a 5 5 0 0 1
##   1b 6 5 0 0 2
##   2a 1 4 5 0 0
##   2b 1 7 4 0 0
##   2c 3 5 1 0 0
##   3a 0 0 3 3 0
##   3b 0 3 1 2 0
##   4a 0 0 3 5 3
##   5a 2 0 0 4 4
##   5b 3 0 0 6 3
```

3

```
statesPlus <- c(1:10)
states <- statesPlus[-10]

# actions = c("stay", "next")

reward <- rep(0,10)
reward[10] <- 1

theta <- 0.1
gamma <- 0.95

V <- rep(0,10)
```

```
repeat{

  delta <- 0
  for(s in states){
    v <- V[s]
    V[s] <- max(reward[s] + gamma*V[s], reward[s+1] + gamma*V[s+1])
    delta <- max(delta, abs(v - V[s]))
  }

  if (delta <theta)
    break
}

policy <- rep(0,9)

for(s in states){
  policy[s] <- as.numeric(reward[s] + gamma*V[s] <= reward[s+1] + gamma*V[s+1])
}

print("Values")
```

```
## [1] "Values"
```

```
V
```

```
##  [1] 0.6634204 0.6983373 0.7350919 0.7737809 0.8145062 0.8573750 0.9025000
##  [8] 0.9500000 1.0000000 0.0000000
```

```r
print("optimal policy")
```

```
## [1] "optimal policy"
```

```r
policy
```

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

The optimal policy is to move to the next state for every state. This is because of that the values increase for each state

4

4.1

Functions from lab

```r
library(kernlab)
#nested Square Exponetial Kernel
nestedSEK <- function(sigmaF=10,l=100) {
  fixedSEK <- function(x1,x2){
    n1 <- length(x1)
    n2 <- length(x2)
    K <- matrix(NA,n1,n2)
    for (i in 1:n2){
      K[,i] <- sigmaF^2*exp(-0.5*( (x1-x2[i])/l)^2 )
    }
    return(K)
  }
  class(fixedSEK) <- 'kernel'
  return(fixedSEK)
}

SEK <- nestedSEK()

nestedPeriodic <- function(sigmaF = 20,l1 =1, l2 = 100, d=365) {
  periodicKernel <- function(x, xstar){
    n1 <- length(x)
    n2 <- length(xstar)
    K <- matrix(NA,n1,n2)

    for (i in 1:n2){
      absDiff <- absDiff <- abs(x-xstar[i])
      K[,i] <- sigmaF^2*exp(-2*sin(pi*absDiff/d)^2/l1^2)*exp(-0.5*absDiff^2/l2^2)
    }
    return(K)
  }
  class(periodicKernel) <- 'kernel'
  return(periodicKernel)
}

periodic <- nestedPeriodic()
```

```r
X <- c(1, 182, 365)
```

```
# kernel matrix where x = X, y = Xstar
kernelMatrix(kernel = SEK, x = X, y = X)
```

```
## An object of class "kernelMatrix"
##             [,1]      [,2]        [,3]
## [1,] 100.0000000  19.43587   0.1327046
## [2,]  19.4358672 100.00000  18.7411227
## [3,]   0.1327046  18.74112 100.0000000
```

```
kernelMatrix(kernel = periodic, x = X, y = X)
```

```
## An object of class "kernelMatrix"
##             [,1]      [,2]        [,3]
## [1,] 400.0000000  10.52494   0.5307397
## [2,]  10.5249423 400.00000  10.1457164
## [3,]   0.5307397  10.14572 400.0000000
```

4.2

Algorithm 2.1

```
posteriorGP <- function(X, y, XStar, sigmaNoise, k, ...){
  n <- length(X)
  K <- k(X, X, ...)
  kStar <- k(X,XStar)

  #Cholesky
  L <- t(chol(K + sigmaNoise^2*diag(n)))

  #alpha
  alpha <- solve(t(L),solve(L,y))

  #Posterior mean = fStar
  kStar <- k(X, XStar)
  fStar <- t(kStar)%*%alpha

  #Posterior variance
  v <- solve(L,kStar)
  variance <- k(XStar, XStar) - t(v)%*%v

  #Marginal log-likelihood log p(y|X)
  log_marg_likelihood <- -0.5*(t(y)%*%alpha)-sum(log(diag(L)))-(n/2)*log(2*pi)

  return(list(mean =fStar, variance =variance, mll = log_marg_likelihood))
}
```

```
tempData <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/TempTulli
tempData <- cbind(tempData, time = 1:nrow(tempData))
tempData <- cbind(tempData, day = ((tempData$time-1)%%365)+1)

#trainData <- subset(tempData, (time - 1)%%5 == 0)



X <- tempData$time
```

```r
Y <- tempData$temp

polyFit <- lm(Y ~  X + I(X^2))
sigmaNoise <- sd(polyFit$residuals)

lmlSEK <- posteriorGP(X, Y, X, sigmaNoise, SEK)
lmlPeridic <- posteriorGP(X, Y, X, sigmaNoise, periodic)

print("Square exponential")
```

```
## [1] "Square exponential"
```

```r
lmlSEK$mll
```

```
##           [,1]
## [1,] -6886.958
```

```r
print("Periodic kernel")
```

```
## [1] "Periodic kernel"
```

```r
lmlPeridic$mll
```

```
##          [,1]
## [1,] -6927.51
```

I would select the square exponential kernel