Project 8

Santiago Castro Dau, June Monge, Rachita Kumar, Sarah Lötscher

2022-05-05

Problem 20: Classical NEMs

1. For each model, construct the transitive closure (by adding edges) and define the corresponding adjacency matrices Φ and Θ, which represent the signalling pathways and the E-gene attachments. Determine the corresponding expected effect patterns (F).

```
phi1 = t(array(c(c(1,0,1,1,1),
                 c(0,1,0,0,1),
                 c(0,0,1,1,1),
                 c(0,0,0,1,1),
                 c(0,0,0,0,1)),
                 dim = c(5, 5), dimnames = list(c("S1", "S2", "S3", "S4", "S5"),
                                               c("S1", "S2", "S3", "S4", "S5"))))

phi2 = t(array(c(c(1,0,0,1,1),
                 c(0,1,0,0,1),
                 c(1,0,1,1,1),
                 c(0,0,0,1,1),
                 c(0,0,0,0,1)),
                 dim = c(5, 5), dimnames = list(c("S1", "S2", "S3", "S4", "S5"),
                                               c("S1", "S2", "S3", "S4", "S5"))))
```

Construct phi

```
theta1 = array(dim = c(5,6), dimnames = list(c("S1", "S2", "S3", "S4", "S5"),
                                             c("E1", "E2", "E3", "E4", "E5","E6")))
theta1["S1",] = c(0,0,0,0,0,0)
theta1["S2",] = c(0,0,0,1,0,1)
theta1["S3",] = c(1,1,0,0,0,0)
theta1["S4",] = c(0,0,1,0,0,0)
theta1["S5",] = c(0,0,0,0,1,0)

theta2 = array(dim = c(5,6), dimnames = list(c("S1", "S2", "S3", "S4", "S5"),
                                             c("E1", "E2", "E3", "E4", "E5","E6")))
theta2["S1",] = c(1,1,0,0,0,0)
theta2["S2",] = c(0,0,0,1,0,1)
```

```r
theta2["S3",] = c(0,0,0,0,0,0)
theta2["S4",] = c(0,0,1,0,0,0)
theta2["S5",] = c(0,0,0,0,1,0)

theta1
```

Construct thetha

```
##    E1 E2 E3 E4 E5 E6
## S1  0  0  0  0  0  0
## S2  0  0  0  1  0  1
## S3  1  1  0  0  0  0
## S4  0  0  1  0  0  0
## S5  0  0  0  0  1  0
```

```r
theta2
```

```
##    E1 E2 E3 E4 E5 E6
## S1  1  1  0  0  0  0
## S2  0  0  0  1  0  1
## S3  0  0  0  0  0  0
## S4  0  0  1  0  0  0
## S5  0  0  0  0  1  0
```

```r


F1 = phi1%*%theta1
F2 = phi2%*%theta2
print("F1")
```

Calculate $F = \Phi\Theta$

```
## [1] "F1"
```

```r
F1
```

```
##    E1 E2 E3 E4 E5 E6
## S1  1  1  1  0  1  0
## S2  0  0  0  1  1  1
## S3  1  1  1  0  1  0
## S4  0  0  1  0  1  0
## S5  0  0  0  0  1  0
```

```r
print("F2")
```

```
## [1] "F2"
```

```r
F2
```

```
##    E1 E2 E3 E4 E5 E6
## S1  1  1  1  0  1  0
## S2  0  0  0  1  1  1
## S3  1  1  1  0  1  0
## S4  0  0  1  0  1  0
## S5  0  0  0  0  1  0
```

2. Assuming no noise, determine the discrete data D1 and D2 from both models. Given only the data, can you tell apart the two models?

```
D1 = array(dim = c(6, 5), dimnames = list(c("E1", "E2", "E3", "E4", "E5","E6"),
                                           c("S1", "S2", "S3", "S4", "S5")))

D1["E1",] = c(1,0,1,0,0)
D1["E2",] = c(1,0,1,0,0)
D1["E3",] = c(1,0,1,1,0)
D1["E4",] = c(0,1,0,0,0)
D1["E5",] = c(1,1,1,1,0)
D1["E6",] = c(0,1,0,0,1)

D2 = array(dim = c(6, 5), dimnames = list(c("E1", "E2", "E3", "E4", "E5","E6"),
                                           c("S1", "S2", "S3", "S4", "S5")))

D2["E1",] = c(1,0,1,0,0)
D2["E2",] = c(1,0,1,0,0)
D2["E3",] = c(1,0,1,1,0)
D2["E4",] = c(0,1,0,0,0)
D2["E5",] = c(1,1,1,1,0)
D2["E6",] = c(0,1,0,0,1)

D1
```

```
##    S1 S2 S3 S4 S5
## E1  1  0  1  0  0
## E2  1  0  1  0  0
## E3  1  0  1  1  0
## E4  0  1  0  0  0
## E5  1  1  1  1  0
## E6  0  1  0  0  1
```

```
D2
```

```
##    S1 S2 S3 S4 S5
## E1  1  0  1  0  0
## E2  1  0  1  0  0
## E3  1  0  1  1  0
## E4  0  1  0  0  0
## E5  1  1  1  1  0
## E6  0  1  0  0  1
```

Since the Data matrices D1 and D2 are identical, we cannot tell the two models apart. ### 3. Use the mnem1 package for this question: Take D1 and D2 from the previous question. For each model, calculate the marginal log-likelihood ratio (network score) given the data by setting the false positive rate to be 5 and the false negative rate to be 1.

```
library(mnem)
```

```
## Registered S3 methods overwritten by 'RcppEigen':
##   method               from
##   predict.fastLm       RcppArmadillo
##   print.fastLm         RcppArmadillo
##   summary.fastLm       RcppArmadillo
##   print.summary.fastLm RcppArmadillo
```

```
#nem1 = nem(D1,marginal = TRUE,fpfn = c(0.05,0.01))
#nem1$score
#nem2 = nem(D2,marginal = TRUE,fpfn = c(0.05,0.01))
#nem2$score
###Not sure which one it is
scoreAdj(D1,adj = phi1,method="disc",marginal=TRUE,fpfn=c(0.05,0.01))$score
```

```
## [1] 52.42384
```

```
scoreAdj(D2,adj = phi2,method="disc",marginal=TRUE,fpfn=c(0.05,0.01))$score
```

```
## [1] 52.42384
```

Problem 21: Hidden Markov NEMs

1. Using the defnitions for HM-NEMs from the lecture, compute the transition probabilities from $G_t = u$ to $G_{t+1} \in v1, v2$ for different smoothness parameter $\lambda \in 0.1, \ldots 0.9$.

```
## not sure if the enumerate has to be used and how?
u = t(array(c(c(1,1,1,0),
              c(0,1,1,1),
              c(0,0,1,1),
              c(0,0,0,1)),
              dim = c(4, 4), dimnames = list(c("S1", "S2", "S3", "S4"),
                                             c("S1", "S2", "S3", "S4"))))
v1 = t(array(c(c(1,1,1,0),
              c(0,1,1,1),
              c(0,0,1,0),
              c(0,0,0,1)),
              dim = c(4, 4), dimnames = list(c("S1", "S2", "S3", "S4"),
                                             c("S1", "S2", "S3", "S4"))))
v2 = t(array(c(c(1,0,0,0),
              c(1,1,1,0),
              c(1,0,1,0),
              c(1,0,0,1)),
              dim = c(4, 4), dimnames = list(c("S1", "S2", "S3", "S4"),
                                             c("S1", "S2", "S3", "S4"))))
```

```r
lambda = seq(0.1, 0.9, by=0.1)

s_uv1 = sum(u!=v1)
s_uv2 = sum(u!=v2)
w = mnem:::enumerate.models(c("S1","S2","S3","S4"),trans.close = FALSE)
```

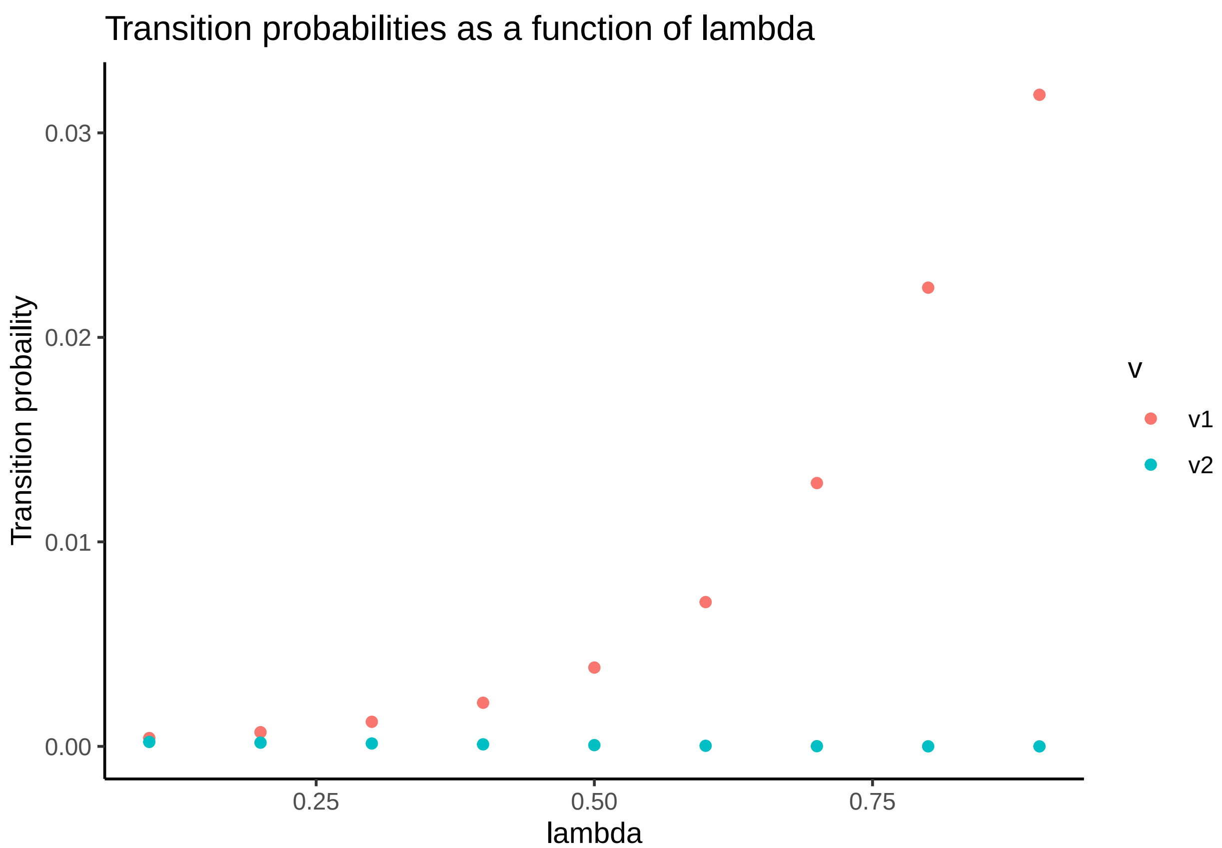
```
## Generated 4096 unique models ( out of 4096 )
```

```r
s_uw = array(dim = c(length(w), 1))
for(i in 1:length(w)){
  s_uw[i] = sum(u!=w[[i]])
}

power <- function(x, y) sign(x) * abs(x)^y


T = array(dim = c(9,2), dimnames = list(lambda,c("v1", "v2")))
C = array(dim = c(9,1), dimnames = list(lambda,c("C")))

for(i in lambda){
  C[as.character(i),] = sum(power((1-i),s_uw)*i)
  T[as.character(i),"v1"] = (1/C[as.character(i),])*((1-i)^s_uv1)*i
  T[as.character(i),"v2"] = (1/C[as.character(i),])*((1-i)^s_uv2)*i
}
T
```

```
##                v1           v2
## 0.1 0.0004066299 2.160998e-04
## 0.2 0.0006915442 1.812842e-04
## 0.3 0.0012014646 1.413511e-04
## 0.4 0.0021316282 9.945325e-05
## 0.5 0.0038536733 6.021365e-05
## 0.6 0.0070554312 2.889905e-05
## 0.7 0.0128765947 9.387038e-06
## 0.8 0.0224313310 1.435605e-06
## 0.9 0.0318630818 3.186308e-08
```

2. Plot the transition probabilities for v_1 and v_2 as a function of λ. Describe the transition probabilities as a function of λ.

```r
library(reshape2)
library(ggplot2)
library(RColorBrewer)
data = data.frame(melt(T))
colnames(data)<-c("lambda","v","T")
plot<-ggplot(data,aes(x=lambda,y=T,color=v))+
  geom_point()+
  theme_classic()+
  ylab("Transition probaility")+
  labs(title="Transition probabilities as a function of lambda")
plot
```



Problem 22: Mixture NEMs

1. Determine the the cellular perturbation map ρ, where $\rho_{ic} = 1$ if cell c is perturbed by a knock-down of S-gene i.

```
rho = array(dim = c(2,4), dimnames = list(c("S1","S2"),
                                          c("C1", "C2", "C3", "C4")))
rho["S1",] = c(1,0,1,0)
rho["S2",] = c(0,1,1,1)
rho
```

```
##    C1 C2 C3 C4
## S1  1  0  1  0
## S2  0  1  1  1
```

2. Assume that C_1, C_2 are generated from F_1 and C_3, C_4 are generated from F_2, compute the noiseless log odds matrix R, where $R_{jc} > 0$ means that the perturbation on cell c has an effect on E-gene j:

```
phi_F1 = array(dim = c(2,2), dimnames = list(c("S1","S2"),
                                             c("S1","S2")))
```

```r
phi_F1["S1",] = c(1,1)
phi_F1["S2",] = c(0,1)

phi_F2 = array(dim = c(2,2), dimnames = list(c("S1","S2"),
                                             c("S1","S2")))
phi_F2["S1",] = c(1,0)
phi_F2["S2",] = c(1,1)

theta_F1 = array(dim = c(2,2), dimnames = list(c("S1","S2"),
                                               c("E1","E2")))
theta_F1["S1",] = c(1,0)
theta_F1["S2",] = c(0,1)

theta_F2 = array(dim = c(2,2), dimnames = list(c("S1","S2"),
                                               c("E1","E2")))
theta_F2["S1",] = c(0,1)
theta_F2["S2",] = c(1,0)

EEP_F1 = t(t(rho)%*%phi_F1%*%theta_F1)
EEP_F1[EEP_F1>1] = 1
EEP_F2 = t(t(rho)%*%phi_F2%*%theta_F2)
EEP_F2[EEP_F2>1] = 1

print("Expected effect pattern of F1")
```

(a) For each component k, compute the expected effect pattern $(\rho^T\phi^k\theta^k)^T$. Replace all non-zeros by 1.

```
## [1] "Expected effect pattern of F1"
```

```r
EEP_F1
```

```
##    C1 C2 C3 C4
## E1  1  0  1  0
## E2  1  1  1  1
```

```r
print("Expected effect pattern of F2")
```

```
## [1] "Expected effect pattern of F2"
```

```r
EEP_F2
```

```
##    C1 C2 C3 C4
## E1  0  1  1  1
## E2  1  1  1  1
```

```r
R= cbind(EEP_F1[,1:2],EEP_F2[,3:4])
R[R==0] = -1
R
```

(b) Based on the component assignment for each cell, extract the corresponding column from the expected effect patterns computed above and put it into R. Replace all zeros by -1.

```
##    C1 C2 C3 C4
## E1  1 -1  1  1
## E2  1  1  1  1
```

3. Take R from the previous question. Given the vector of mixture weights $\pi = (0.44, 0.56)$, calculate the responsibilities Γ. Then, update the mixture weights.

```r
library("expm")
```

```
## Loading required package: Matrix

##
## Attaching package: 'expm'

## The following object is masked from 'package:Matrix':
##
##     expm
```

```r
L1 = t(EEP_F1)%*%R
L2 = t(EEP_F2)%*%R
print("L1")
```

```
## [1] "L1"
```

```r
L1
```

```
##    C1 C2 C3 C4
## C1  2  0  2  2
## C2  1  1  1  1
## C3  2  0  2  2
## C4  1  1  1  1
```

```r
print("L2")
```

```
## [1] "L2"
```

```r
L2
```

```
##    C1 C2 C3 C4
## C1  1  1  1  1
## C2  2  0  2  2
## C3  2  0  2  2
## C4  2  0  2  2
```

```r
pi = c(0.44,0.56)

gamma = array(dim = c(2,4), dimnames = list(c("F1","F2"),
                                            c("C1", "C2", "C3", "C4")))

gamma["F1",] = pi[1]*exp(diag(L1))/(pi[1]*exp(diag(L1))+pi[2]*exp(diag(L2)))
gamma["F2",] = pi[2]*exp(diag(L2))/(pi[2]*exp(diag(L2))+pi[1]*exp(diag(L1)))

##Responsibilities should be in  [0,1]??
print("Responsibilities")
```

```
## [1] "Responsibilities"
```

```r
gamma
```

```
##           C1        C2   C3        C4
## F1 0.6811014 0.6811014 0.44 0.2242338
## F2 0.3188986 0.3188986 0.56 0.7757662
```

```r
pi[1] = sum(gamma["F1",])/(sum(gamma["F1",])+sum(gamma["F2",]))
pi[2] = sum(gamma["F2",])/(sum(gamma["F1",])+sum(gamma["F2",]))

print("Updated mixture weights")
```

```
## [1] "Updated mixture weights"
```

```r
pi
```

```
## [1] 0.5066091 0.4933909
```