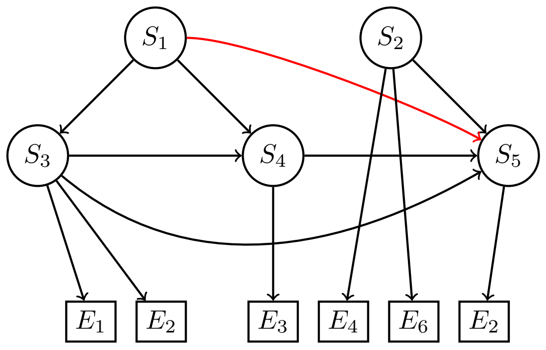
Project 8

Richard Affolter

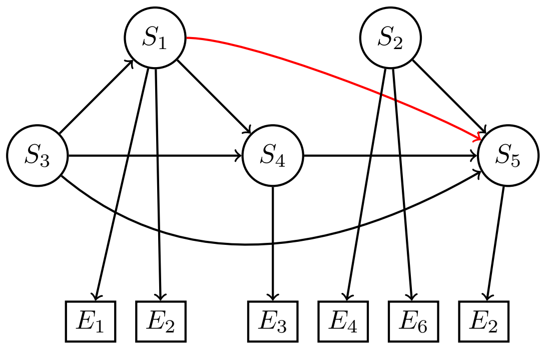
27 April, 2021

Difficulty of this project: **3**

Problem 20: Classical NEMs



(a)



(b)

1. For each model, identify the transitive 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).

For both models the transitive edges are added in red.

For model (a) we have the following adjacency matrix Φ and E-gene attachments Θ

```
library(mnem)
Phi_a_open = rbind(
  c(1,0,1,1,0),
  c(0,1,0,0,1),
  c(0,0,1,1,1),
  c(0,0,0,1,1),
  c(0,0,0,0,1)
  )

# compute transitive closure
Phi_a = transitive.closure(Phi_a_open)

colnames(Phi_a) = rownames(Phi_a) = paste0("S",1:5)
Phi_a
```

```
##    S1 S2 S3 S4 S5
## S1  1  0  1  1  1
## S2  0  1  0  0  1
## S3  0  0  1  1  1
## S4  0  0  0  1  1
## S5  0  0  0  0  1
```

```
Theta_a = rbind(
  c(0,0,0,0,0,0),
  c(0,0,0,1,0,1),
  c(1,1,0,0,0,0),
  c(0,0,1,0,0,0),
  c(0,0,0,0,1,0)
)
colnames(Theta_a) = paste0("E",1:6)
rownames(Theta_a) = paste0("S",1:5)

Theta_a
```

```
##    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
```

By multiplying Φ and Θ we get F.

```
F_a = Phi_a %*% Theta_a
F_a
```

```
##    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
```

For model (b) we have the following adjacency matrix Φ, E-gene attachments Θ and therefore F:

```
Phi_b_open = rbind(
  c(1,0,0,1,0),
  c(0,1,0,0,1),
  c(1,0,1,1,1),
  c(0,0,0,1,1),
  c(0,0,0,0,1)
  )

# compute transitive closure
Phi_b = transitive.closure(Phi_b_open)

colnames(Phi_b) = rownames(Phi_b) = paste0("S",1:5)
Phi_b
```

```
##    S1 S2 S3 S4 S5
## S1  1  0  0  1  1
## S2  0  1  0  0  1
## S3  1  0  1  1  1
## S4  0  0  0  1  1
## S5  0  0  0  0  1
```

```
Theta_b = rbind(
  c(1,1,0,0,0,0),
  c(0,0,0,1,0,1),
  c(0,0,0,0,0,0),
  c(0,0,1,0,0,0),
  c(0,0,0,0,1,0)
)
colnames(Theta_b) = paste0("E",1:6)
rownames(Theta_b) = paste0("S",1:5)

Theta_b
```

```
##    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
```

```
F_b = Phi_b %*% Theta_b
F_b
```

```
##    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 D_1 and D_2 from both models. Given only the data, can you tell apart the two models?

With perfect data the discrete data is equal to the transposed expected effect patterns ($F_a^T = D_1$, $F_b^T = D_2$). We can quickly see that the data is the same, and we therefore cannot tell the two models apart.

```
D_1 = t(F_a)
D_2 = t(F_b)

D_1
```

```
##    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  1
## E6   0  1  0  0  0
```

```
D_2
```

```
##     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  1
## E6   0  1  0  0  0
```

```
all.equal(D_1, D_2)
```

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

3. Use the `mnem` package for this question: Take D_1 and D_2 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%.

```
score_1 = mnem::scoreAdj(D = D_1, adj = Phi_a, method = "disc",
                         marginal = TRUE, fpfn = c(0.05, 0.01),
                         logtype = exp(1))$score

score_2 = mnem::scoreAdj(D = D_2, adj = Phi_b, method = "disc",
                         marginal = TRUE, fpfn = c(0.05, 0.01),
                         logtype = exp(1))$score

cat("network score for D_1: ", score_1, "\n")
```

```
## network score for D_1:  42.07706
```

```
cat("network score for D_2: ", score_2, "\n")
```

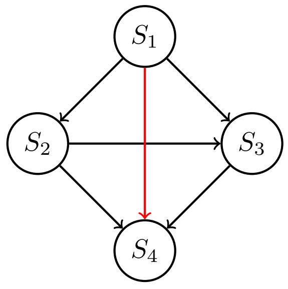
```
## network score for D_2:  42.07706
```

As we can see both networks have the same score.

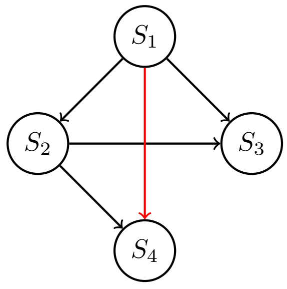
Problem 21: Hidden Markov NEMs

The transitive edges are added in red.

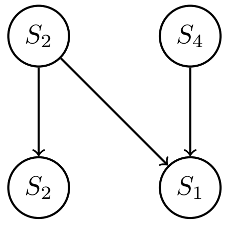
Using the definitions for HM-NEMs from the lecture, compute the transition probabilities from $G_t = u$ to $G_{t+1} \in \{v_1, v_2\}$ for different smoothness parameter $\lambda \in \{0.1, ..., 0.9\}$.



u



v_1



v_2

```
# construct the transitively closed graph adj matrices
u_open = rbind(
  c(1,1,1,0),
  c(0,1,1,1),
  c(0,0,1,1),
  c(0,0,0,1)
)

u = transitive.closure(u_open)
colnames(u)= rownames(u) = paste0("S",1:4)
u
```

```
##    S1 S2 S3 S4
## S1  1  1  1  1
## S2  0  1  1  1
## S3  0  0  1  1
## S4  0  0  0  1
```

```
v_1_open = rbind(
  c(1,1,1,0),
  c(0,1,1,1),
  c(0,0,1,0),
  c(0,0,0,1)
)

v_1 = transitive.closure(v_1_open)
colnames(v_1)= rownames(v_1) = paste0("S",1:4)
v_1
```

```
##    S1 S2 S3 S4
## S1  1  1  1  1
## S2  0  1  1  1
## S3  0  0  1  0
## S4  0  0  0  1
```

```
v_2_open = rbind(
  c(1,0,0,0),
  c(1,1,1,0),
  c(1,0,1,0),
  c(1,0,0,1)
)

v_2 = transitive.closure(v_2_open)
colnames(v_2)= rownames(v_2) = paste0("S",1:4)
v_2
```

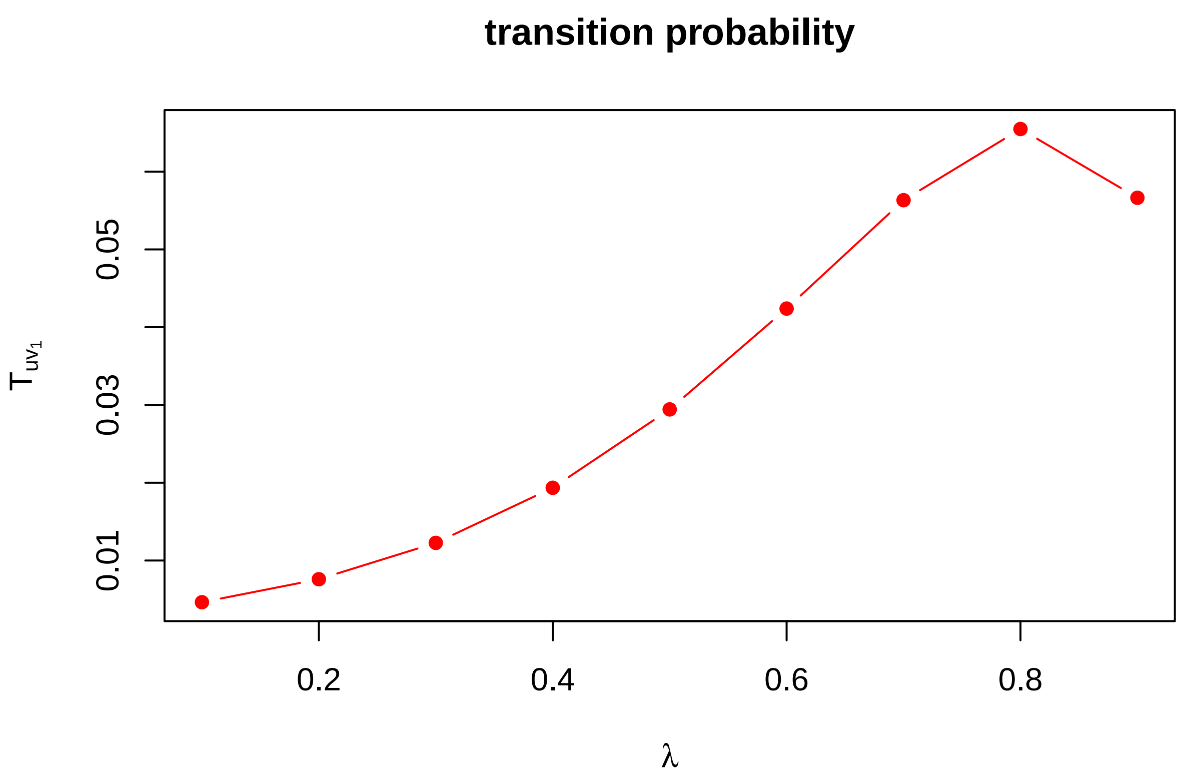
```
##    S1 S2 S3 S4
## S1  1  0  0  0
## S2  1  1  1  0
## S3  1  0  1  0
## S4  1  0  0  1
```

```
library(dplyr)
get_unnormalized_trans_prob = function(u, v, lambda){
  suv = sum(u != v)
  return((1-lambda)^(suv)*lambda)
}

get_normalizing_constant = function(lambda){
  model_space = mnem:::enumerate.models(4, verbose = FALSE, trans.close = TRUE)
  sapply(model_space, get_unnormalized_trans_prob, u=u, lambda=lambda) %>%
    sum() %>% return()
}
lambdas = seq(0.1, 0.9, by=0.1)
C_us = sapply(lambdas, get_normalizing_constant)

Tuv_1 = sapply(lambdas, get_unnormalized_trans_prob, u=u, v=v_1)/C_us
Tuv_2 = sapply(lambdas, get_unnormalized_trans_prob, u=u, v=v_2)/C_us

toprint = rbind(lambdas, Tuv_1, Tuv_2)
rownames(toprint) = c("lambda", "Transition u -> v1", "Transition u -> v2")
toprint
```

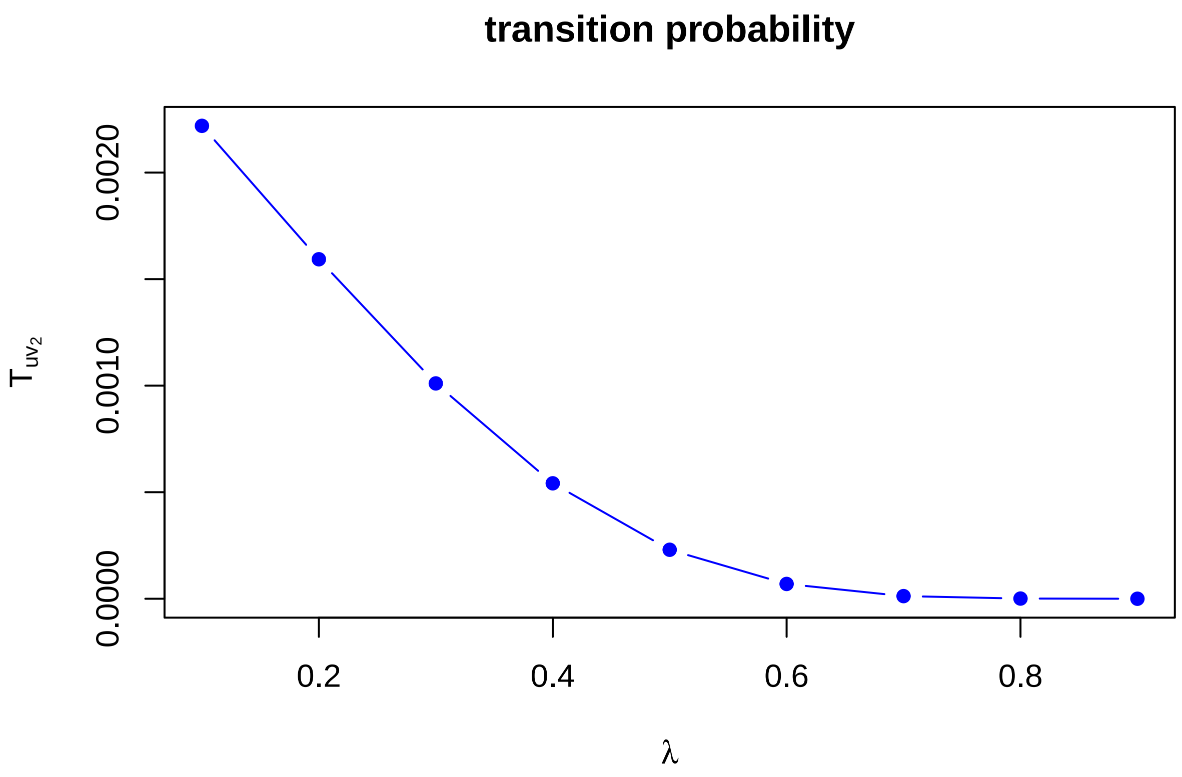
```
##                           [,1]        [,2]        [,3]         [,4]
## lambda             0.100000000 0.200000000 0.300000000 0.4000000000
## Transition u -> v1 0.004640039 0.007597382 0.012270196 0.0193532270
## Transition u -> v2 0.002219316 0.001593287 0.001010503 0.0005417665
##                            [,5]         [,6]         [,7]         [,8]
## lambda             0.5000000000 6.000000e-01 7.000000e-01 8.000000e-01
## Transition u -> v1 0.0294282471 4.239384e-02 5.632821e-02 6.547811e-02
## Transition u -> v2 0.0002299082 6.945807e-05 1.231898e-05 8.381199e-07
##                            [,9]
## lambda             9.000000e-01
## Transition u -> v1 5.663736e-02
## Transition u -> v2 5.663736e-09
```

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
plot(lambdas, Tuv_1, xlab = expression(lambda), ylab = expression(T[uv[1]]),
     type = "b", col = "red", main = "transition probability", pch=16)
```

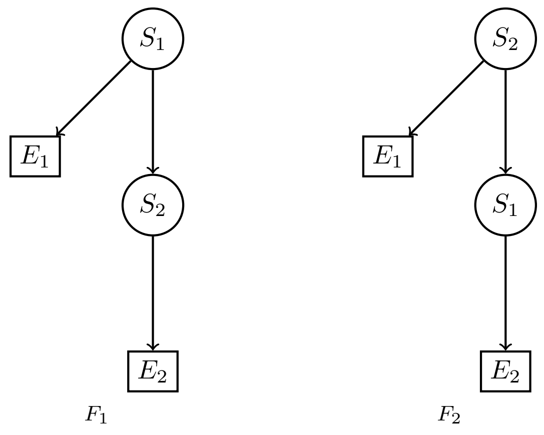


```r
plot(lambdas, Tuv_2, xlab = expression(lambda), ylab = expression(T[uv[2]]),
     type = "b", col = "blue", main = "transition probability", pch=16)
```



We can see that the transition probability T_{uv_1} increases with λ up to a certain point. The opposite effect is happening with T_{uv_2}, which decreases with λ. This can be explained by considering that network v_1 is very close to u, there is only one edge missing. On the contrary Network v_2 is very different to u. This means that $s_{uv_1} = 1$ and $s_{uv_2} = 8$. If λ is close to 1, then $(1-\lambda)^{s_{uv}}$ trends to zero very fast. So a high smoothness parameter rewards transitions to similar networks and punishes transitions to diverse networks.

Problem 22: Mixture NEMs



Given are two NEMs F_1 and F_2 with two S-genes $\{S_1, S_2\}$ and two E-genes $\{E_1, E_2\}$. The data contains four cells $\{C_1, C_2, C_3, C_4\}$. $\{C_1, C_3\}$ are perturbed by a knock-down of S_1, and $\{C_2, C_3, C_4\}$ are perturbed by a knock-down of S_2.

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 = rbind(
  c(1,0,1,0),
  c(0,1,1,1)
)
colnames(rho) = paste0("C",1:4)
rownames(rho) = paste0("S",1:2)
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:

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

```
phi_1 = rbind(
  c(1,1),
  c(0,1)
  )
colnames(phi_1) = rownames(phi_1) = paste0("S",1:2)

phi_2 = rbind(
  c(1,0),
  c(1,1)
  )

colnames(phi_1) = rownames(phi_1) = paste0("S",1:2)
colnames(phi_2) = rownames(phi_2) = paste0("S",1:2)

theta_1 = rbind(
  c(1,0),
  c(0,1)
)

theta_2 = rbind(
  c(0,1),
  c(1,0)
)

colnames(theta_1) = paste0("S",1:2)
rownames(theta_1) = paste0("E", 1:2)

colnames(theta_2) = paste0("S",1:2)
rownames(theta_2) = paste0("E", 1:2)
```

```
# compute expected effect pattern
expected_1 = (t(rho) %*% phi_1 %*% theta_1) %>% t()
expected_2 = (t(rho) %*% phi_2 %*% theta_2) %>% t()

rownames(expected_1) = rownames(theta_1)
rownames(expected_2) = rownames(theta_2)

# Replace all non-zeros by 1
expected_1[which(expected_1 != 0)] = 1
expected_2[which(expected_2 != 0)] = 1

expected_1
```

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

```
expected_2
```

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

(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.

```
R = cbind(expected_1[,c("C1","C2")], expected_2[,c("C3","C4")])

# Replace all zeros by -1
R[which(R == 0)] = -1
R
```

```
##    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.

```
# calculate the log likelihood ratio
L_1 = t(expected_1) %*% R
L_2 = t(expected_2) %*% R

# calculate mixture weights
Lk = list(L_1, L_2)
pi = c(k1=0.44, k2=0.56)

get_unnormalized_weight = function(pi, L){
  return(pi * exp(diag(L)))
}

# get unnormalized, transposed gamma
gamma = mapply(get_unnormalized_weight, pi, Lk)

# normalize and transpose to get gamma
gamma = (gamma/rowSums(gamma)) %>% t()

gamma
```

```
##           C1        C2   C3        C4
## k1 0.6811014 0.6811014 0.44 0.2242338
## k2 0.3188986 0.3188986 0.56 0.7757662
```

```
#update the mixture weights
pi = rowSums(gamma) / (sum(gamma))

pi
```

```
##        k1        k2
## 0.5066091 0.4933909
```