Chapter 15: Temporal Bayesian networks

W45-Wed
- No class Fri or Mon
- Midterm grades are out

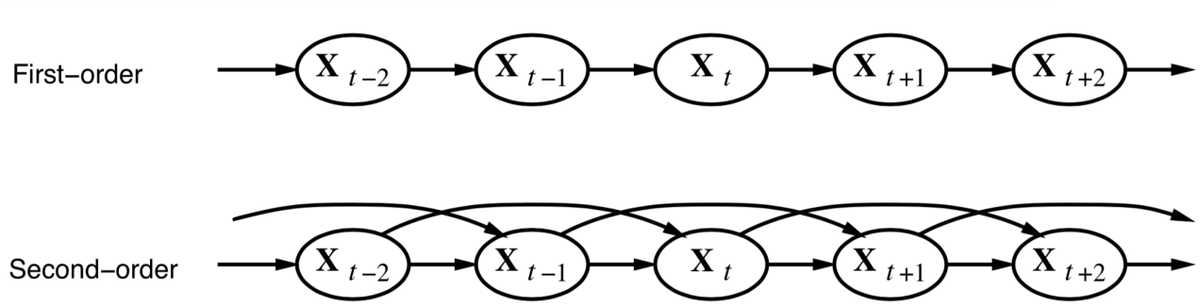
The world changes; we need to track and predict it
Examples:
- Tracking an object's trajectory: military, ecology, ...
- Disease management
- Economics/finance

Basic idea: copy state and evidence variables for each time step
X_t = set of unobservable state variables at time t e.g., BloodSugart, StomachContentst, etc.
E_t = set of observable evidence variables at time t
e.g., MeasuredBloodSugart, PulseRatet, FoodEatent

This assumes discrete time; step size depends on problem Notation: Xa:b =Xa,Xa+1,...,Xb−1,Xb

Markov process

Markov assumption: X_t depends on a fixed subset of $X_{0:t}$

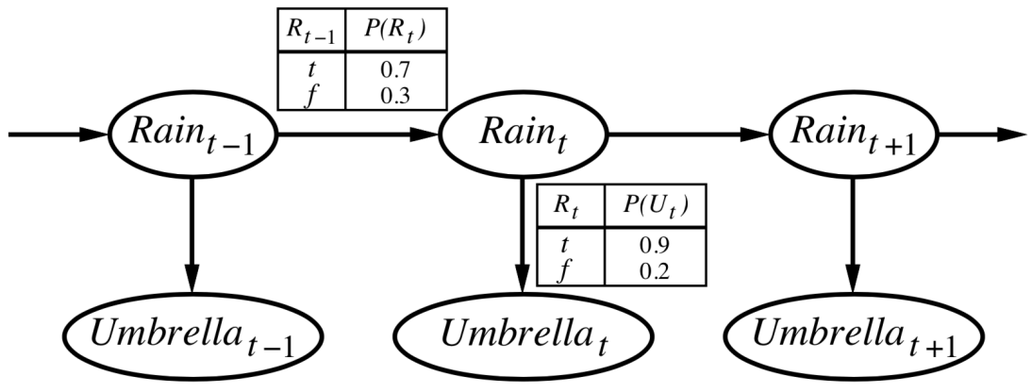


First-order Markov process: P(XtlX0:t–1) = P(XtlXt–1)
Second-order Markov process: P(XtlX0:t–1) = P(XtlXt–2, Xt–1)

Markov assumption is never strictly true. Options for improving accuracy of model:

- Increase order of Markov process
- Add more variables. Ex: velocity.

Example: Umbrella and rain



We're a dog that never goes outside. We see if our owner takes their umbrella each day. We want to know if it is raining.

Naive inference

Naive algorithm: Use inference by enumeration.

Inference tasks

Filtering: $P(X_t|e_{1:t})$

Prediction: $P(X_{t+k}|e_{1:t})$

Smoothing: $P(X_k|e_{1:t})$

Most likely explanation: $\text{argmax}_{x_{1:t}} P(x_{1:t}|e_{1:t})$

W46-Wed:

- A3 due next Monday
- Look at your midterms in office hours.
-

Filtering

Goal: Find a recursive state estimation algorithm. We want:
P(X_t+1 | e_1:t+1) = f(e_t+1, P(X_t | e_1:t))

P(X_t+1 | e_1:t, e_t+1)
= 1/Z P(X_t+1, e_t+1 | e_1:t)
= 1/Z sum_xt P(X_t+1, xt, e_1:t, e_t+1)
= 1/Z P(e_t+1 | X_t+1) sum_Xt P(X_t, e+1:t) P(X_t+1 | X_t)

Algorithm: For i=1...t, compute P(X_i | e_1:i)

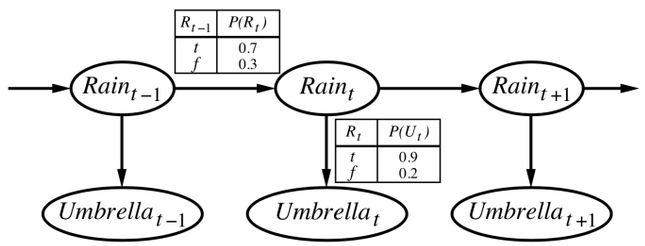
Filtering

```
def forward(e, T, HMM):
    f = matrix(T, HMM.D)
    f[0,:] = HMM.P(X_0)
    for t in 1:T:
        for i in 1:D:
            for j in 1:D:
                f[t,i] += HMM.P(e_t | X_t+1 = i) *
                          HMM.P(X_t+1 = i | X_t = j) *
                          f[t-1, j]
    return f[T, :] / sum(f[T, :])
```

e: evidence
T: Length of sequence.
HMM.D: Size of the domain of the hidden variable (e.g. D=2 for the rain example).
HMM.P: Conditional probability distribution of the HMM.

Filtering example

Observed data: [umbrella, umbrella, no umbrella]



f:
t=0: 0.5 0.5
t=1:
  R: 0.9*(0.5*0.7 + 0.5*0.3) = 0.45
  -R: 0.2*(0.5*0.3 + 0.5*0.7) = 0.1
t=2:
  R: 0.9*(0.45*0.7 + 0.1*0.3) = 0.31
  -R: 0.2*(0.45*0.3 + 0.1*0.7) = 0.04
t=3:
  R: 0.1*(0.31*0.7 + 0.04*0.3) = 0.023
  -R: 0.8*(0.31*0.3 + 0.04*0.7) = 0.097
alpha = 1/0.12
output: [0.192, 0.808]

Smoothing

$$
\begin{aligned}
\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{e}_{1:k}) \\
&= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) \\
&= \alpha\mathbf{f}_{1:k}\mathbf{b}_{k+1:t}
\end{aligned}
$$

Backwards probability:

$$
\begin{aligned}
\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) &= \textstyle\sum_{\mathbf{x}_{k+1}}\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \textstyle\sum_{\mathbf{x}_{k+1}}P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \textstyle\sum_{\mathbf{x}_{k+1}}P(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)
\end{aligned}
$$

We can divide the problem into two subproblems.

P(X_k I e:t) = P(X_k I e1:k, ek+1:t)
= α P(X_k I e_1:k) P(e_k+1:t I X_k, e_1:k)
= αP(X_k I e_1:k) P(e_k+1:t I X_k)
= α f_1:k * b_k+1:t

P(e_{k+1:t} I X_k)
= sum_{x_k+1} P(e_k+1:t I x_k+1) P(x_k+1IX_k)
= sum_{x_k+1} P(e_k+1 I x_k+1) P(e_k+2:t I x_k+1) P(x_k+1IX_k)

Smoothing

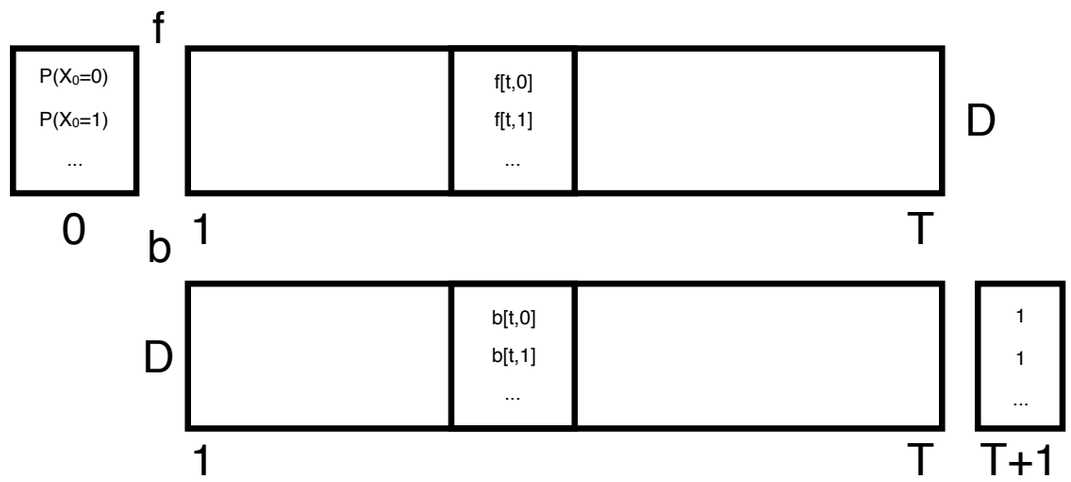
$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\Sigma_{\mathbf{x}_t}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{e}_{1:t})$$

$$\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) = \Sigma_{\mathbf{x}_{k+1}}P(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$$

Summary of algorithm:

- Store both f and b; each a matrix of T x D.
- Initialize: f[0,:] = P(X_0). b[T+1,:] = vector of 1's.
- Iteratively compute f[i,:] using f[i-1,:]. Likewise compute b[i,:] using b[i+1,:]
- Compute P(X_i | e_1:t) = alpha * f[i,:] * b[i,:]

Smoothing



alpha = 1 /(f[t,0]*b[t,0] + f[t,1]*b[t,1])

$P(X_t=1 \mid e_{1:T})$ = alpha * f[t,1] * b[t,1]

Viterbi

Most likely explanation: $\text{argmax}_{x_{1:t}} P(x_{1:t}|e_{1:t})$

m(x_t) <- prob of most likely sequence of states ending in x_t
m(x_t) = max_{x_1:t-1} P(x_t, x_{1:t-1} | e_1:t)
= P(e_{t+1} | X_t+1) max_{x_t}

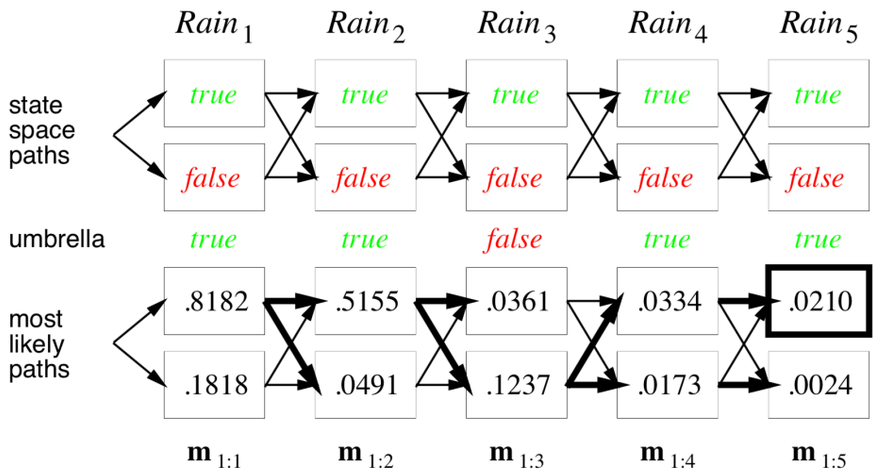
Most likely sequence != sequence of most likely states!

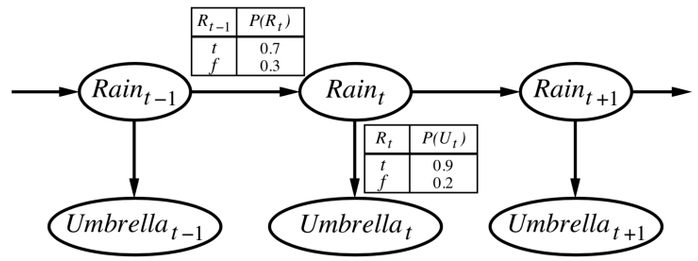
argmax: Returns the item with the highest value. (In contrast to max, which returns the value itself).
e.g.:
argmax({"a": 1, "b": 2}) -> "b"
max({"a": 1, "b": 2}) -> 2

Filtering

```
def viterbi(e, T, HMM):
    m = matrix(T, HMM.D)
    prev = matrix(T, HMM.D)
    m[0,:] = HMM.P(X_0)
    for t in 1:T:
        for i in 1:HMM.D:
            prob = list[j]
            for j in 1:D:
                prob[j] = m[t-1, j] *
                          HMM.P(i | j) *
                          HMM.P(e_t | i)
            m[t,i] = max(prob)
            prev[t,i] = argmax(prob)
    path = list(T)
    path[T] = argmax[m[T,:]]
    for t in T-1:1:
        path[t] = prev[t+1, path[t+1]]
    return path
```

Viterbi example





Using probabilities in log space

$$\log(x \cdot y) = \log(x) + \log(y)$$

$$\log(x + y) = \log(x) + \log(1 + \exp(\log(y) - \log(x))$$

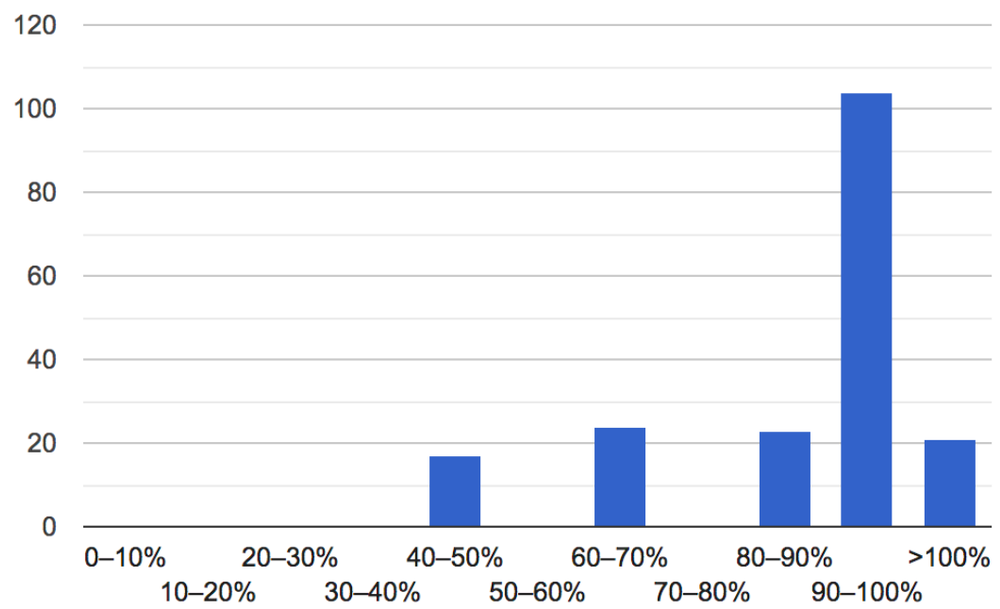
Probabilities often run into underflow.

x*y => log(x * y) = log(x) + log(y)
x + y =>
Bad way: log(exp(x) + exp(y))
Good way: log(x+y) = log(x) + log(1 + exp(log(y) - log(x))
Choose x to be the larger (less negative) of the two.
implemented by log_sum() in A3

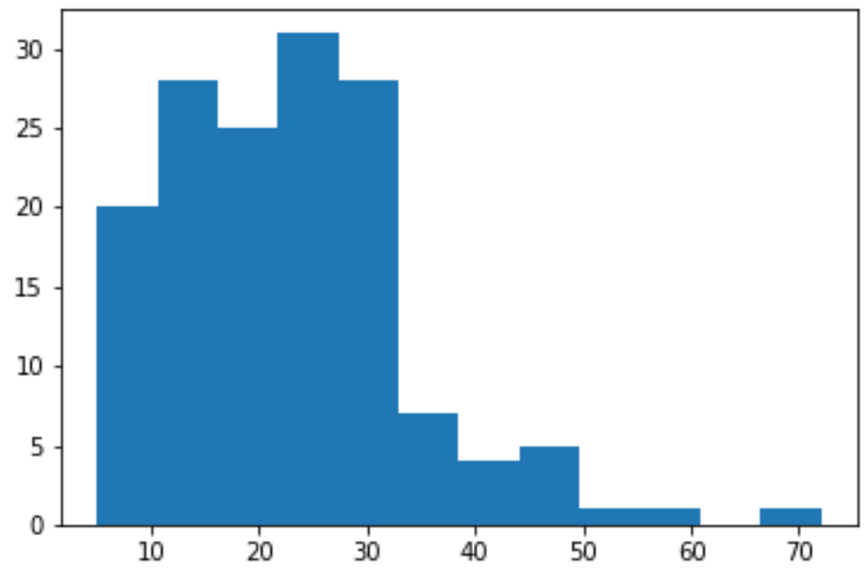
Assignment 3

- You will implement both Viterbi and smoothing.
- Posterior decode: argmax_{x_k} P(x_k | e_{i:t}). Provided for you.
- Use probabilities in log space.

Assignment 2 recap: grade distribution



Assignment 2 recap: time spent



Assignment 2 recap

ID Number	Solved instance size	
301338190	365	used a modern DPLL Algorithm with "Trail"
301290598	105	Optimized Unit Propagation, improved variable slection heuristic
301381172	100	effcient backtracking ;improved variable slection heuristic
301357131	95	improved variable slection heuristic
301254694	90	improved variable slection heuristic
301383936	90	
301338480	90	
301386155	85	
301263491	80	
301249689	75	
301093642	75	
301321221	75	
301308729	75	
301310345	75	
301354426	75	
301344819	70	
301300876	70	
301279383	65	
301127666	65	
301287252	65	
301070053	60	

Problem 14.1

We have a bag of three biased coins a,b and c with probabilities of coming up heads of 20%, 60%, and 80%, respectively. One coin is drawn randomly from the bag (with equal likelihood of drawing each of the three coins), and then the coin is flipped three times to generate the outcomes X1, X2, and X3.

(a) Draw the Bayesian network corresponding to this setup and define the necessary CPTs.

(b) Calculate which coin was most likely to have been drawn from the bag if the observed flips come out heads twice and tails once.

Problem 14.1

a. With the random variable C denoting which coin $\{a, b, c\}$ we drew, the network has C at the root and X_1, X_2, and X_3 as children.

The CPT for C is:

C	$P(C)$
a	$1/3$
b	$1/3$
c	$1/3$

The CPT for X_i given C are the same, and equal to:

C	X_1	$P(C)$
a	$heads$	0.2
b	$heads$	0.6
c	$heads$	0.8

b. The coin most likely to have been drawn from the bag given this sequence is the value of C with greatest posterior probability $P(C|2 \text{ heads, } 1 \text{ tails})$. Now,

$$\begin{aligned} P(C|2 \text{ heads, } 1 \text{ tails}) &= P(2 \text{ heads, } 1 \text{ tails}|C)P(C)/P(2 \text{ heads, } 1 \text{ tails}) \\ &\propto P(2 \text{ heads, } 1 \text{ tails}|C)P(C) \\ &\propto P(2 \text{ heads, } 1 \text{ tails}|C) \end{aligned}$$

where in the second line we observe that the constant of proportionality $1/P(2 \text{ heads, } 1 \text{ tails})$ is independent of C, and in the last we observe that $P(C)$ is also independent of the value of C since it is, by hypothesis, equal to $1/3$.

From the Bayesian network we can see that X_1, X_2, and X_3 are conditionally independent given C, so for example

$$\begin{aligned} &P(X_1 = tails, X_2 = heads, X_3 = heads|C = a) \\ &= P(X_1 = tails|C = a)P(X_2 = heads|C = a)P(X_3 = heads|C = a) \\ &= 0.8 \times 0.2 \times 0.2 = 0.032 \end{aligned}$$

Note that since the CPTs for each coin are the same, we would get the same probability above for any ordering of 2 heads and 1 tails. Since there are three such orderings, we have

$$P(2heads, 1tails|C = a) = 3 \times 0.032 = 0.096.$$

Similar calculations to the above find that

$$\begin{aligned} P(2heads, 1tails|C = b) &= 0.432 \\ P(2heads, 1tails|C = c) &= 0.384 \end{aligned}$$

showing that coin b is most likely to have been drawn.

Alternatively, one could directly compute the value of $P(C|2 \text{ heads, } 1 \text{ tails})$.

Problem 14.16

Investigate the complexity of exact inference in general Bayesian networks. Prove that any 3-SAT problem can be reduced to exact inference in a Bayesian network constructed to represent the particular problem and hence that exact inference is NP-hard.

(Hint: consider a network with one variable for each proposition symbol, one for each clause, and one for the conjunction of clauses.)

Problem 14.16

a. Consider a 3-CNF formula $C_1 \wedge \ldots C_n$ with n clauses where each clause is a disjunct $C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$ of literals i.e., each ℓ_{ij} is either P_k or $\neg P_k$ for some atomic proposition $P_1, \ldots, P_m$.

Construct a Bayesian network with a (boolean) variable S for the whole formula, C_i for each clause, and P_k for each atomic proposition. We will define parents and CPTs such that for any assignment to the atomic propositions, S is true if and only if the 3-CNF formula is true.

Atomic propositions have no parents, and are true with probability 0.5. Each clause C_i has as its parents the atomic propositions corresponding to the literals ℓ_{i1}, ℓ_{i2}, and ℓ_{i3}). The clause variable is true iff one of its literals is true. Note that this is a deterministic CPT. Finally, S has all the clause variables C_i as its parents, and if true if any only if all clause variables are true.

Notice that $P(S = True) > 0$ if and only if the formula is satisfiable, and exact inference will answer this question.

Problem 15.13

A professor wants to know if students are getting enough sleep. Each day, the professor observes whether the students sleep in class, and whether they have red eyes. The professor has the following domain theory:

- The prior probability of getting enough sleep, with no observation, is 0.7.
- The probability of getting enough sheep on night t is 0.8 given that the student got enough sleep the previous night, and 0.3 if not.
- The probability of having red eyes is 0.2 if the student got enough sleep, and 0.7 if not.
- The probability of sleeping in class is 0.1 if the student got enough sleep, and 0.3 if not.

Formulate this information as a dynamic Bayesian network that the professor could use to filter or predict form a sequence of observations. Then reformulate it as a hidden Markov model that has only a single observation variable. Five the complete probability tables for the model.

Problem 15.13

15.13 The DBN has three variables: S_t, whether the student gets enough sleep; R_t, whether they have red eyes in class; C_t, whether the student sleeps in class. S_t is a parent of S_{t+1}, R_t, and C_t. The CPTs are given by

$$\begin{aligned} P(s_0) &= 0.7 \\ P(s_{t+1}|s_t) &= 0.8 \\ P(s_{t+1}|\neg s_t) &= 0.3 \\ P(r_t|s_t) &= 0.2 \\ P(r_t|\neg s_t) &= 0.7 \\ P(c_t|s_t) &= 0.1 \\ P(c_t|\neg s_t) &= 0.3 \end{aligned}$$

To reformulate as an HMM with a single observation node, simply combine the 2-valued variables "having red eyes" and "sleeping in class" into a single 4-valued variable, multiplying together the emission probabilities. (Probability tables omitted.)