Informatics 2D. Tutorial 9
Probabilities and Bayesian Networks Continued

Week 10

Part 1: Approximate Inference in Bayesian Networks

This part is based on the car model from last week (Figure 1):

1. Explain the basic principle used in *rejection sampling*, and apply it to select which of the following samples is used by the algorithm for the query:

   $$P(D|R = true, L = false)$$

   Describe the main problem with using *rejection sampling*.

d	e	a	b	r	l	
T	F	T	F	F	F	
F	F	T	F	T	F	
F	T	F	T	T	T	
T	F	T	T	T	F	
F	F	F	T	F	F	

2. Using Algorithm 1, create a sample and compute its weight for the query:

   $$P(D|R = true, L = false)$$

   How does it avoid the problem seen in the rejection algorithm?

$P(d) = 0.75$

$P(e) = 0.95$

d	e	$P(a)$
T	T	0.9
T	F	0.3
F	T	0.1
F	F	0.1

a	$P(b)$
T	0.8
F	0.2

b	$P(l)$
T	0.9
F	0.05

b	$P(r)$
T	0.8
F	0.1

Figure 1: Probabilities for the car model

Algorithm 1 Likelihood Weighting algorithm

```
function LIKELIHOOD-WEIGHTING($X$,$\mathbf{e}$,$bn$,$N$) returns an estimate of $P(X|\mathbf{e})$
inputs:
        $X$, the query variable
        $\mathbf{e}$, observed values for variables $\mathbf{E}$
        $bn$, a Bayes net with variables $\{\mathbf{X}\} \cup \mathbf{E} \cup \mathbf{Y}$    /*$\mathbf{Y}$=hidden vars*/
        $N$, the total number of samples to be generated
local variables: $\mathbf{W}$, a vector of weighted counts over $X$, initially 0

for $j = 1$ to $N$ do
  $\mathbf{x}, w \leftarrow$ WEIGHTED-SAMPLE($bn$,$\mathbf{e}$)
  $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where $x$ is the value of $X$ in $\mathbf{x}$
return Normalize($\mathbf{W}[X]$)

function WEIGHTED-SAMPLE($bn$,$\mathbf{e}$) returns an event and a weight
$\mathbf{x} \leftarrow$an event with $n$ elements; $w \leftarrow 1$
for $i$=1 to $n$ do
  if $X_i$ has a value $x_i$ in $\mathbf{e}$
    then $w \leftarrow w \times P(X_i = x_i | parents(X_i))$
    else $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i|\text{parents}(X_i))$
return $\mathbf{x}$,$w$
```

Solutions

1. The Rejection-Sampling algorithm generates samples from the prior distribution specified for the network. Then it rejects all those that do not match the evidence.

   In this case, the evidence is $\neg l \wedge r$, and therefore all the sample that do not have $R = True$ and $L = False$ are rejected.

d	e	a	b	r	l	
T	F	T	F	F	F	✗
F	F	T	F	T	F	✓
F	T	F	T	T	T	✗
T	F	T	T	T	F	✓
F	F	F	T	F	F	✗

   From Russell & Norvig

   > The biggest problem with rejection sampling is that it rejects so many samples. The fraction of samples consistent with the evidence **e** drops exponentially as the number of evidence variables grows, so the procedure is unusable for complex problems.

2. Set the weight to 1.0
   Sample from $P(D) = \langle 0.75, 0.25 \rangle$, suppose it returns *false*.
   Sample from $P(E) = \langle 0.95, 0.05 \rangle$, suppose it returns *true*.
   Sample from $P(A|D = false, E = true) = \langle 0.1, 0.9 \rangle$, suppose it returns *true*.
   Sample from $P(B|A = true) = \langle 0.8, 0.2 \rangle$, suppose it returns *true*.

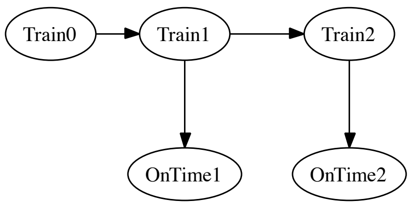

Figure 2: Task DBN

Tr_{t-1}	$P(Tr_t)$
t	0.8
f	0.9

Table 1: Transition model

Tr_t	$P(On_t)$
t	0.8
f	0.1

Table 2: Sensor model

L is an evidence variable with value $false$. Therefore we set:
$w \leftarrow w \times P(L = false|B = true) = 0.1$
R is an evidence variable with value $True$. Therefore we set:
$w \leftarrow w \times P(R = true|B = true) = 0.08$
So we obtain a sample $[false, true, true, true, false, true]$ with weight 0.08 under $D = false$.

From Russell & Norvig:

> Likelihood weighting avoids the inefficiency of rejection sampling by generating only events that are consistent with the evidence **e**.

Part 2: Inference in Temporal Models

Task description

Living close to his office, John has no information about the punctuality of trains. But looking at Tim, who arrives every morning by train, John can guess if the train had a delay or not. Tim might be late for other reasons, so John cannot be sure about the trains. However, Tim is usually punctual and can be considered a good indicator.

Dynamic Bayesian Network

A dynamic Bayesian network is a Bayesian network that represents a temporal probability distribution. In general, each slice of a DBN can have any number of state variables $\mathbf{X}_t$ and evidence variables $\mathbf{E}_t$. For simplicity, we will assume that the variables and their links are exactly replicated from slice to slice and that the DBN represents a first order Markov process, so that each variable can have parents only in its own slice of the immediatly preceding slice.

The DBN used to model the punctuality of trains is in figure 2.

Inference in Temporal Models

Computing the posterior probability of an event of the past given evidence up to the present is called *smoothing* or *hindsight*, that is $\mathbf{P}(\mathbf{X}_k|e_{1:t})$ for $1 \leq k < t$. $\mathbf{P}(\mathbf{X}_k|e_{1:t})$ is most conveniently computed in two parts: the evidence up to k and the evidence from $k+1$ to t.

$$\begin{aligned}
&\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) = \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{e}_{1:k}) \qquad \textit{using Bayes' rule} \\
&= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) \qquad \textit{using conditional independence} \\
&= \alpha f_{1:k} b_{k+1:t}
\end{aligned}$$

Therefore, $f_{1:k} = \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})$ is the *forward message* from $t = 1$ to $t = k$, while $b_{k+1:t} = \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k)$ is the backward message from the current time t back to $t = k + 1$.

1. Explain the meaning of the terms in the following formulas. How do you obtain the numbers?

   (a) $f_{1:t+1} = \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{e}_{1:t})$

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

Smoothing exercise

Given the list of observations $On = [on_1 = \textit{true}, on_2 = \textit{false}]$, and knowing that $\mathbf{P}(Tr_0) = \langle 0.7, 0.3\rangle$, we will compute the posterior distribution of the probability of Tr_1.

First of all, to compute general expression $P(X_k|e_{1:t})$ in order to obtain the smoothed estimate for the probability that the train is late at $t = 1$, given the observations about punctuality on days 1 and 2, means to compute $P(Tr_1|on_1, \neg on_2)$.
We have seen earlier that:

$$\mathbf{P}(\mathbf{X}_k|e_{1:t}) = \alpha\mathbf{f}_{1:k}\mathbf{b}_{k+1:t} = \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k)$$

therefore:

$$P(Tr_1|on_1, \neg on_2) = \alpha\mathbf{f}_{1:k}\mathbf{b}_{k+1:t} = \alpha\mathbf{P}(Tr_1|on_1)\mathbf{P}(\neg on_2|Tr_1) \tag{1}$$

The term $\mathbf{f}_{1:k} = \mathbf{P}(Tr_1|on_1)$ is computed as follows. On day 1, Tim is on time, so $On_1 = true$. Prediction from day 0 to day 1 is:
$\mathbf{P}(Tr_1) = \sum_{tr_0} \mathbf{P}(Tr_1|tr_o)P(tr_0)$

$= \langle 0.8, 0.2\rangle \times 0.7 + \langle 0.9, 0.1\rangle \times 0.3$
$= \langle 0.56, 0.14\rangle + \langle 0.27, 0.03\rangle$
$= \langle 0.83, 0.17\rangle$
Updating the value with evidence for $t = 1$:

$\mathbf{P}(Tr_1|on_1) = \alpha \mathbf{P}(on_1|Tr_1)\mathbf{P}(Tr_1) = \alpha \langle 0.8, 0.1\rangle \langle 0.83, 0.17\rangle$

$= \alpha \langle 0.664, 0.017\rangle = \frac{1}{0.664+0.01} \langle 0.664, 0.017\rangle = \langle 0.975,\ 0.025\rangle$
We have obtained the first term in the equation 1.

2. Compute the term $\mathbf{b}_{k+1:t}$ in equation 1.

3. Plug the results into the equation 1 and compute the smoothed estimate for the train being late on the second day

Hidden Markov Model

Every hidden Markov model can be represented as a DBN with a *single state variable* and a *single evidence variable.* Every discrete variable DBN can be represented as an HMM: we can combine all the state variables in the DBN into a single state variable whose values are all the possible tuples of the individual state variables.

If we consider the single, discrete state variable X_t has values denoted by integers $1, \dots, S$, where S is the number of possible states, then the *transition model* $\mathbf{P}(X_t|X_{t-1})$ becomes an $S \times S$ matrix $\mathbf{T}$, where $\mathbf{T}_{ij} = P(X_t = j|X_{t-1} = i)$ and the index i indicates the row, while j indicates the column.

The *sensor model* can also be put in matrix form: for each time step t, we construct a diagonal matrix $\mathbf{O}_t$ whose diagonal entries are given by the values $P(e_t|X_t = i)$ and whose other entries are 0.
The forward equation, seen above, becomes:

$$\mathbf{f}_{1:t+1} = \mathbf{O}_{t+1}\mathbf{T}^T\mathbf{f}_{1:t}$$

where $\mathbf{T}^T$ is the transposed matrix of $\mathbf{T}$. The backward equation becomes:

$$\mathbf{b}_{k+1:t} = \mathbf{T}\mathbf{O}_{k+1}\mathbf{b}_{k+2:t}$$

4. Is the model in the example an HMM? Why?

5. Write the $\mathbf{T}$ matrix for the example.

6. Write the $\mathbf{O}_1$and $\mathbf{O}_2$ matrices for the example (given that $on_1 = true$ and $on_2 = false$)

7. Given the same situation seen in question 2 ($\mathbf{On} = [on_1 = true, on_2 = false]$, $\mathbf{P}(Tr_0) = \langle 0.7, 0.3\rangle$), compute the posterior distribution of the probability of Tr_1 using the matrix formulation for the forward and backward equations:

   (a) compute $\mathbf{f}_{1:k}$ with the matrices.

   (b) compute $\mathbf{b}_{k+1:t}$ with the matrices.

   (c) compute $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) = \alpha \mathbf{f}_{1:k}\mathbf{b}_{k+1:t}$.

Solutions

1. (a) α is the normalisation factor.
   $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})$ is obtained from the *sensor model.*
   Within the summation:
   $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t)$ is obtained from the *transition model.*
   $\mathbf{P}(\mathbf{x}_t|\mathbf{e}_{1:t})$ is obtained from the current state distribution, and is the *recursive term.*
   The process is:
   $\mathbf{f}_{1:t+1} = \alpha Forward(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$
   where $Forward$ is $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{e}_{1:t})$

   (b) $P(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})$ is obtained from the *sensor model.*
   $P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})$ is the *recursive term.*
   $\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$ is obtained from the *transition model.*

   The process is:
   $b_{k+1:t} = Backward(b_{k+2:t}, e_{k+1:t})$
   where $Backward$ is in $\sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}|\mathbf{x}_{k+1}) P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$.

2. To compute $b_{k+1:t} = \mathbf{P}(\neg on_2|Tr_1)$:
   $\mathbf{P}(\neg on_2|Tr_1) = \sum_{tr_2} P(\neg on_2|tr_2) P(|tr_2) \mathbf{P}(tr_2|Tr_1)$
   where $P(\neg on_2|tr_2)$ is 0.2 (from sensor model), and $P(|tr_2)$ is 1 by definition, and $P(tr_2|Tr_1)$ is given by the transition model and is $\langle 0.8, 0.9\rangle$ for tr_2 and $\langle 0.2, 0.1\rangle$ for $\neg tr_2$
   $= (0.2 \times 1 \times \langle 0.8, 0.9\rangle) + (0.9 \times 1 \times \langle 0.2, 0.1\rangle)$
   $= \langle 0.16, 0.18\rangle + \langle 0.18, 0.09\rangle = \langle 0.34, 0.27\rangle$.

3. $P(Tr_1|on_1, \neg on_2) = \alpha \mathbf{f}_{1:k} \mathbf{b}_{k+1:t} = \alpha \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k}) \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k)$
   $= \alpha\langle 0.664, 0.017\rangle \times \langle 0.34, 0.27\rangle = \alpha\langle 0.22576, 0.00459\rangle =$
   $= \frac{1}{0.22576+0.00459}\langle 0.22576, 0.00459\rangle = \langle 0.98, 0.02\rangle$.

4. Yes, because it has a single state variable, $Train_t$, and a single evidence variable, $OnTime_t$.

5. Setting $i = 1$ for $X_{t-1} = true$, $i = 2$ for $X_{t-1} = false$ and $j = 1$ for $X_t = true$ and $j = 2$ for $X_t = false$, and using $\mathbf{T}_{ij} = P(X_t = j|X_{t-1} = i)$

$$\mathbf{T} = \begin{pmatrix} P(x_t|x_{t-1}) & P(\neg x_t|x_{t-1}) \\ P(x_t|\neg x_{t-1}) & P(\neg x_t|\neg x_{t-1}) \end{pmatrix} = \begin{pmatrix} 0.8 & 0.2 \\ 0.9 & 0.1 \end{pmatrix}$$

6.

$$\mathbf{O}_1 = \begin{pmatrix} P(On_1 = true|Tr_1 = true) & 0 \\ 0 & P(On_1 = true|Tr_1 = false) \end{pmatrix} = \begin{pmatrix} 0.8 & 0 \\ 0 & 0.1 \end{pmatrix}$$

$$\mathbf{O}_2 = \begin{pmatrix} P(On_2 = false|Tr_2 = true) & 0 \\ 0 & P(On_2 = false|Tr_2 = false) \end{pmatrix} = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.9 \end{pmatrix}$$

7. (a) $\mathbf{f}_{1:k} = \mathbf{f}_{1:1} = \alpha \mathbf{O}_1 \mathbf{T}^T \mathbf{f}_0 = \alpha \begin{pmatrix} 0.8 & 0 \\ 0 & 0.1 \end{pmatrix} \begin{pmatrix} 0.8 & 0.2 \\ 0.9 & 0.1 \end{pmatrix}^T \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix} =$
$\alpha \begin{pmatrix} 0.8 & 0 \\ 0 & 0.1 \end{pmatrix} \begin{pmatrix} 0.8 & 0.9 \\ 0.2 & 0.1 \end{pmatrix} \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 0.975 \\ 0.025 \end{pmatrix}$

   (b) $\mathbf{b}_{k+1:t} = \mathbf{b}_{2:2} = \mathbf{T}\mathbf{O}_2 \mathbf{b}_{3:2} = \begin{pmatrix} 0.8 & 0.2 \\ 0.9 & 0.1 \end{pmatrix} \begin{pmatrix} 0.2 & 0 \\ 0 & 0.9 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} =$
$\begin{pmatrix} 0.34 \\ 0.27 \end{pmatrix}$
The same result as using the first method.

   (c) $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) = \alpha \mathbf{f}_{1:k} \mathbf{b}_{k+1:t}$ becomes:
$P(Tr_1|on_1, \neg on_2) = \alpha \begin{pmatrix} 0.975 \\ 0.025 \end{pmatrix} \begin{pmatrix} 0.34 \\ 0.27 \end{pmatrix} = \begin{pmatrix} 0.98 \\ 0.0199 \end{pmatrix}$
Again, the same result.

Matrix and Vector revision

Vectors

- A vector is an ordered sequence of values: $\mathbf{x} = \langle 3, 4 \rangle$, $\mathbf{y} = \langle 2, 3 \rangle$.
- Vector addition $\mathbf{x} + \mathbf{y}$ is the elementwise sum: $\mathbf{x} + \mathbf{y} = \langle 3 + 2, 4 + 3 \rangle = \langle 5, 7 \rangle$
- Scalar multiplication multiplies each element by a constant: $2\mathbf{x} = \langle 2 \times 3, 2 \times 4 \rangle = \langle 6, 8 \rangle$
- The *dot product* of vector is obtained as:
  $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$
  So, in the example:
  $\mathbf{x} \cdot \mathbf{y} = \langle 3 * 2, 4 * 3 \rangle = \langle 6, 12 \rangle$

Matrices

- A matrix is a rectangular array of values arranged into rows and columns:
  $\mathbf{M} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}$ The index i indicates the row, while j indicates the column.
- The sum of two matrices is defined by adding corresponding elements:
  $\mathrm{M} + \mathrm{N} = m_{ij} + n_{ik}$
  The sum is undefined if M and N have different sizes.
  For example:
  $$\mathbf{M} + \mathbf{N} = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} + \begin{pmatrix} 3 & 4 \\ 5 & 2 \end{pmatrix} = \begin{pmatrix} 1+3 & 2+4 \\ 2+5 & 3+2 \end{pmatrix} = \begin{pmatrix} 4 & 6 \\ 7 & 5 \end{pmatrix}$$
- The multiplication of a matrix by a scalar is:
  $c\mathrm{M} = cm_{ij}$
  For example:
  $$cM = 2 \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 4 & 6 \end{pmatrix}$$
- The product between matrices $\mathbf{MN}$ is defined only if the second matrix has the same number of rows as the first has columns:
  if M is of size $a \times b$, then N must be of size $b \times c$, and resulting matrix is of size $a \times c$.
  If the matrices are of appropriate size, then:
  $\mathbf{MN} = \sum_j m_{ij} n_{jk}$
  In the example:
  $$M + N = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 3 & 4 \\ 5 & 2 \end{pmatrix} = \begin{pmatrix} 1 \times 3 + 2 \times 5 & 1 \times 4 + 2 \times 2 \\ 2 \times 3 + 3 \times 5 & 2 \times 4 + 3 \times 2 \end{pmatrix} = \begin{pmatrix} 13 & 8 \\ 21 & 14 \end{pmatrix}$$
- The transpose of a matrix $\mathbf{M}$ is written $\mathbf{M}^T$ and is formed turned rows into columns. For example:
  $$N^T = \begin{pmatrix} 3 & 4 \\ 5 & 2 \end{pmatrix}^T = \begin{pmatrix} 3 & 5 \\ 4 & 2 \end{pmatrix}$$
- The identity matrix I has the elements $I_{i,j}$ equal to 1 when $i = j$ (that is, on the diagonal) and equal to 0 otherwise. The multiplication of a matrix $\mathbf{M}$ by an identiy matrix $\mathbf{I}$ yields the matrix $\mathbf{M}$: $\mathbf{MI} = \mathbf{M}$.