Graphical Models: Assignment 4

I-Horng Huang, Tomas Jakab, Antonio Remiro, Nicholas Williams

16 January 2017

Contributions

i-horng.huang.16@ucl.ac.uk: 12.2, 27.9, 27.11
tomas.jakab.16@ucl.ac.uk: 12.2, 27.10
antonio.remiro.16@ucl.ac.uk: 12.2, 12.4, 12.6, 23.11, 27.6
nicholas.williams.16@ucl.ac.uk: 12.2, 12.6

Exercise 12.2

1. The weight vector $\mathbf{w}$ can be set by maximum likelihood. Consider differentiating $p(y_{1:T}|\mathbf{x}_{1:T}, \mathbf{w})$ wrt $\mathbf{w}$:

$$\frac{d}{d\mathbf{w}}(p(y_{1:T}|\mathbf{x}_{1:T},\mathbf{w})) = \frac{d}{d\mathbf{w}}(\prod_{t=2}^{T}\mathcal{N}(y_t|\mathbf{w}^T\mathbf{x}_{t-1},\sigma_t^2)) = \prod_{t=2}^{T}\frac{1}{\sqrt{2\pi\sigma_t^2}}\frac{d}{d\mathbf{w}}\exp(\sum_{t=2}^{T}-\frac{1}{2\sigma_t^2}(y_t-\mathbf{w}^T\mathbf{x}_{t-1})^2)$$
$$= \Big(\prod_{t=2}^{T}\frac{1}{\sqrt{2\pi\sigma_t^2}}\Big(-\sum_{t=2}^{T}\frac{(y_t-\mathbf{w}^T\mathbf{x}_{t-1})(\overrightarrow{1}^T\mathbf{x}_{t-1})}{\sigma_t^2}\Big)\Big)\Big(\exp\Big(\sum_{t=2}^{T}\frac{-(y_t-\mathbf{w}^T\mathbf{x}_{t-1})^2}{2\sigma^2}\Big)\Big).$$

Setting $\frac{d}{d\mathbf{w}}(p(y_{1:T}|\mathbf{x}_{1:T},\mathbf{w}))$ to zero results in the expression,

$$\sum_{t=2}^{T}\frac{\overrightarrow{1}^T\mathbf{x}_{t-1}\mathbf{x}_{t-1}^T}{\sigma_t^2}\mathbf{w} = \sum_{t=2}^{T}\frac{\overrightarrow{1}^T\mathbf{x}_{t-1}y_t}{\sigma_t^2}, \qquad \text{(i.e. } \mathbf{a}^T\mathbf{w} = b\text{).}$$

Therefore, any weight vector $\mathbf{w}$ set to satisfy this ML condition ($\mathbf{a}^T\mathbf{w} = b$ for that $b \in \mathbb{R}$, $\mathbf{a} \in \mathbb{R}^K$) will be optimal, with $\mathbf{w}_{opt} = \text{argmax}_{\mathbf{w}} p(y_{1:T}|\mathbf{x}_{1:T}, \mathbf{w})$.

2. The hierarchical Bayesian network for this model is presented in Figure 1.

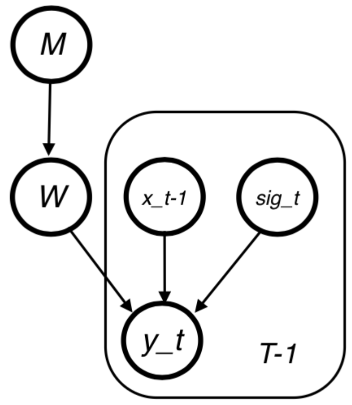

Figure 1: Hierarchical Bayesian network for the model. There is a typo above, there should be T repeats. Alternatively, for $T-1$, x_t, σ_t and y_{t+1} notation.

A given model is indexed by $M = 1, ..., 2^K - 1$. Bayes Theorem gives us,

$$p(M|\mathcal{D}) = \frac{p(M)p(\mathcal{D}|M)}{p(\mathcal{D})},$$

with historical information $\mathcal{D} = \{\mathbf{x}_t, y_{t+1}, t = 1, ..., T-1\}$. We are using a flat prior $p(M) = p(\mathbf{x}_1, ...\mathbf{x}_{T-1}) = C$, a constant. Then,

$$p(M|\mathcal{D}) = \frac{Cp(\mathbf{x}_1, ..., \mathbf{x}_{T-1})p(y_2, ..., y_T|\mathbf{x}_1, ...\mathbf{x}_{T-1}, M)}{p(\mathcal{D})},$$

and lets denote $\frac{Cp(\mathbf{x}_1,...,\mathbf{x}_{T-1})}{p(\mathcal{D})} = c$, also constant over all M models. We have,

$$p(M|\mathcal{D}) = cp(y_2, ..., y_T|\mathbf{x}_1, ...\mathbf{x}_{T-1}, M).$$

Bayesian Model Selection: Involves maximising $p(M|\mathcal{D})$ over all M i.e. maximising $p(y_2, ..., y_T|\mathbf{x}_1, ...\mathbf{x}_{T-1}, M)$. Using $p(y|\mathbf{x}, M) = \int_{\mathbf{w}} p(y|\mathbf{w}, \mathbf{x}, M)p(\mathbf{w}|M)$ and expression (12.9.3),

$$\begin{aligned} p(y_2, ..., y_T|\mathbf{x}_1, ..., \mathbf{x}_{T-1}, M) &= \int_{\mathbf{w}} p(\mathbf{w}|M) \prod_{t=1}^{T-1} p(y_{t+1}|\mathbf{w}, x_t, M) \\ &= \int_{\mathbf{w}} \mathcal{N}(\mathbf{w}|\mathbf{0}_M, \mathbf{I}_M) \prod_{t=1}^{T-1} \mathcal{N}(y_{t+1}|\mathbf{w}^T\mathbf{x}_{t(M)}, \sigma_{t+1}^2). \end{aligned}$$

for a given model M. Then,

$$p(y_2, ..., y_T|\mathbf{x}_1, ...\mathbf{x}_{T-1}, M) = \int_{\mathbf{w}} \frac{1}{(2\pi)^{m/2}} \exp(-\frac{\mathbf{w}^T\mathbf{w}}{2}) \frac{1}{(2\pi)^{\frac{T-1}{2}}} \Big(\prod_{t=1}^{T-1} \frac{1}{|\sigma_{t+1}|^{-1}} \Big) \exp\Big(-\frac{1}{2} \sum_{t=1}^{T-1} \frac{(y_{t_1} - \mathbf{w}^T\mathbf{x}_{i(M)})^2}{\sigma_t + 1} \Big).$$

Set $\mathbf{A} = \mathbf{I}_M + \sum_{t=1}^{T-1} \frac{\mathbf{x}_M\mathbf{x}_M^T}{\sigma_{t+1}^2}$, $\mathbf{b} = \frac{\sum_{t=1}^{T-1} y_{t+1}\mathbf{x}_T}{\sigma_{t+1}^2}$.
Note maximising the expression above is equivalent to maximising $2\log p(y_2, ..., y_T|\mathbf{x}_1, ..., \mathbf{x}_{T-1}, M)$. Observe (12.4.7) in BRML. We obtain a variant,

$$2\log p(y_2, ..., y_T|\mathbf{x}, ..., \mathbf{x}_{T-1}, M) = -(T-1)\log(2\pi) - \sum_{t=1}^{T-1} \frac{y_{t+1}^2}{\sigma_{t+1}^2} + \mathbf{b}^T\mathbf{A}^{-1}\mathbf{b} - \log\det(\mathbf{A}) - \sum_{t=1}^{T-1} \log(\sigma_{t+1}^2).$$

The obtained expression differs from that in (12.4.7) in that α is set to 1 and the volatility is time dependent. In practice, therefore, we compute the expression above for each model M and select the one that maximises it. The selected model is then run for given historical information $\mathcal{D}$.

3. The following code is implemented.

```
max_prob = -10000000;     % initialise min probability
K=6; % binary combinations

% data
dodder = load('dodder.mat')
% dodder = load('./BRMLtoolkit/data/dodder.mat');
sig = dodder.sigma;
X = dodder.x;
Y = dodder.y;
T = dodder.T;
X = X(:,1:T-1);
Y = Y(2:T);
sig = sig(2:T);

% loop over binary combinations
for i =1:(2^K -1)
    binvect=de2bi(i,K); % converts decimal to binary
    dim = sum(binvect); % dimensions
    % for each combination, relevant rows computed.
    pastindices = [];
    for k = 1:K
        if binvect(k)
            pastindices = vertcat(pastindices, k);
        end
    end
    pastX = X(pastindices,:);
    % commputations
    A = eye(dim);
    b = zeros(dim,1);
    for t = 1:T-1
        b = b + (pastX(:,t)*Y(t)/sig(t).^2);
        A = A + (pastX(:,t)*pastX(:,t)')/(sig(t).^2);
    end
    % sums
    bin(i,:) = binvect;
    probability_log(i)= -(T-1)*log(2*pi)-sum((Y.^2)./(sig.^2)) + b'*(A\b) ...
                        -log(det(A)) - sum(log(sig.^2));
    % max log prob update
    if  probability_log(i) > max_prob
        best_bin = binvect;
        max_prob = probability_log(i);
    end

end
best_bin
```

The output of the code is as follows:

```
best_bin =

     1     1     1     1     0     0
```

This indicates that the factors x_1, x_2, x_3, x_4 are those more likely to explain the data.

Exercise 12.4

Are the three people essentially in agreement? To infer this, compare H_{indep} against H_{same}. Based on the given table, the count vector for person 1 is [13, 3, 4], for person 2, [4,9,7], and for person 3, [8,8,4]. For the categorical distribution, given a uniform beta prior on counts and assuming no prior preference for either hypothesis, we have the posterior Bayes' factor,

$$\begin{aligned}\frac{p(\text{persons 1,2 and 3 classify differently})}{p(\text{persons 1, 2 and 3 classify the same})} &= \frac{p(H_{indep}|\mathbf{o}_1,\mathbf{o}_2,\mathbf{o}_3)}{p(H_{same}|\mathbf{o}_1,\mathbf{o}_2,\mathbf{o}_3)} \\ &= \frac{Z(\mathbf{u}+\#^1)Z(\mathbf{u}+\#^2)Z(\mathbf{u}+\#^3)}{Z(\mathbf{u})Z(\mathbf{u}+\#^1+\#^2+\#^3)} \\ &= \frac{Z([14,4,5])Z([5,10,8])Z([9,9,5])}{Z([1,1,1])Z([26,21,16])} \\ &= 2.75 \quad \text{(3 s.f)},\end{aligned}$$

where $Z(\mathbf{u}) = \frac{\prod_{q=1}^{Q}\Gamma(u_q)}{\Gamma(\sum_{q=1}^{Q}u_q)}$. A value of 2.75 indicates anecdotal evidence that the three people are classifying the images differently. There is no strong evidence to believe the data have been generated by different categorical distributions.

Exercise 12.6

(1) We have three parameters in this question, $\theta_{y=1}$, $\theta_{x=1|y=0}$ and $\theta_{x=1|y=1}$, corresponding to three degrees of freedom of $p(x,y)$. The equation for $p(D|M_y y \to x)$ can be reformulated:

$$p(D|M_{y\to x}) = \int_{\theta_{x=1|y=1}}\int_{\theta_{x=1|y=0}}\int_{\theta_{y=1}} p(\theta_{x=1|y=1},\theta_{x=1|y=0})p(\theta_{y=1})\prod_n p(x^n|y^n,\theta_{x=1|y=1},\theta_{x=1|y=0})p(y^n|\theta_y)d\theta_{x=1|y=1}d\theta_{x=1|y=0}d\theta_{y=1}.$$

We can see in the question that $p(\theta_{x=1|y=1}, \theta_{x=1|x=0}) = p(\theta_{x=1|y=1})p(\theta_{x=1|y=0})$ (12.9.18). Hence, the previous equation can be reformulated as:

$$p(D|M_{y\to x}) = \int_{\theta_{x=1|y=1}} p(\theta_{x=1|y=1}) \prod_{n|y=1} p(x^n|y^n = 1, \theta_{x=1|y=1}) d\theta_{x=1|y=1} \times$$
$$\int_{\theta_{x=1|y=0}} p(\theta_{x=1|y=0}) \prod_{n|y=0} p(x^n|y^n = 0, \theta_{x=1|y=0}) d\theta_{x=1|y=0} \times \int_{\theta_{y=1}} p(\theta_{y=1}) \prod_n p(y^n|\theta_y) d\theta_{y=1}.$$

Lets consider each term in the product above individually. Each term can be simplified subbing in the terms corresponding to the priors. For the first term,

$$\int_{\theta_{x=1|y=1}} \frac{\theta_{x=1|y=1}^{\alpha_{x=1|y=1}-1}(1-\theta_{x=1|y=1})^{\beta_{x=1|y=1}-1}\theta_{x=1|y=1}^{\#(x=1,y=1)}(1-\theta_{x=1|y=1})^{\#(x=0,y=1)}}{B(\alpha_{x=1|y=1}, \beta_{x=1|y=1})} d\theta_{x=1|y=1}$$
$$= \frac{1}{B(\alpha_{x=1|y=1}, \beta_{x=1|y=1})} \int_{\theta_{x=1|y=1}} \theta_{x=1|y=1}^{(\#(x=1,y=1)+\alpha_{x=1|y=1}-1)}(1-\theta_{x=1|y=1})^{(\#(x=0,y=1)+\beta_{x=1|y=1}-1)}$$
$$= \frac{B(\#(x=1, y=1) + \alpha_{x=1|y=1}, \#(x=0, y=1) + \beta_{x=1|y=1})}{B(\alpha_{x=1|y=1}, \beta_{x=1|y=1})}.$$

Similarly, for the second term:

$$\int_{\theta_{x=1|y=0}} \frac{\theta_{x=1|y=0}^{\alpha_{x=1|y=0}-1}(1-\theta_{x=1|y=0})^{\beta_{x=1|y=0}-1}\theta_{x=1|y=0}^{\#(x=1,y=0)}(1-\theta_{x=1|y=0})^{\#(x=0,y=0)}}{B(\alpha_{x=1|y=0}, \beta_{x=1|y=0})} d\theta_{x=1|y=0}$$
$$= \frac{1}{B(\alpha_{x=1|y=0}, \beta_{x=1|y=0})} \int_{\theta_{x=1|y=0}} \theta_{x=1|y=0}^{(\#(x=1,y=0)+\alpha_{x=1|y=0}-1)}(1-\theta_{x=1|y=0})^{(\#(x=0,y=0)+\beta_{x=1|y=0}-1)}$$
$$= \frac{B(\#(x=1, y=0) + \alpha_{x=1|y=0}, \#(x=0, y=0) + \beta_{x=1|y=0})}{B(\alpha_{x=1|y=0}, \beta_{x=1|y=0})}.$$

For the third term:

$$\int_{\theta_{y=1}} \frac{\theta_y^{\alpha-1}(1-\theta_y)^{\beta-1}\theta_y^{\#(y=1)}(1-\theta_y)^{\#(y=0)}}{B(\alpha,\beta)} d\theta_{y=1}$$
$$= \frac{B(\#(y=1)+\alpha, \#(y=0)+\beta)}{B(\alpha,\beta)}.$$

The multiplication of the three simplified terms gives the required expression.

(2) We simply have to exchange the positions of x and y. The likelihood of the data is now,

$$p(\mathcal{D}|M_{x\to y}) = \int_\theta p(\theta_{y|x})p(\theta_x) \prod_n p(y^n|x^n, \theta_{y|x})p(x^n|\theta_x).$$

Considering each term in the product above individually; each can be simplified subbing in the terms corresponding to the priors. We assume that $\alpha_{x|y} = \alpha_{y|x}$ and use the notation $\alpha_{x=1|y=1} = \alpha_{1|1}$ for simplicity. For the first term, as in the procedure for part (1), we obtain:

$$\frac{B(\#(x=1, y=1) + \alpha_{1|1}, \#(x=1, y=0) + \beta_{1|1})}{B(\alpha_{1|1}, \beta_{1|1})}$$

For the second term:

$$\frac{B(\#(x=0, y=1) + \alpha_{1|0}, \#(x=0, y=0) + \beta_{1|0})}{B(\alpha_{1|0}, \beta_{1|0})}$$

For the third term:

$$\frac{B(\#(x=1)+\alpha, \#(x=0)+\beta)}{B(\alpha,\beta)}.$$

The multiplication of these three simplified terms gives the required expression for the model with the edge direction reversed.

(3) Dividing the terms obtained for **(1)** by those obtained for **(2)**, we obtain:

$$\frac{p(D|M_{y\to x})}{p(D|M_{x\to y})} = \frac{B(\#(x=1,y=1)+\alpha_{1|1}, \#(x=0,y=1)+\beta_{1|1})}{B(\#(x=1,y=1)+\alpha_{1|1}, \#(x=1,y=0)+\beta_{1|1})} \times$$

$$\frac{B(\#(x=1,y=0)+\alpha_{1|0}, \#(x=0,y=0)+\beta_{1|0})}{B(\#(x=0,y=1)+\alpha_{1|0}, \#(x=0,y=0)+\beta_{1|0})} \times \frac{B(\#(y=1)+\alpha, \#(y=0)+\beta)}{B(\#(x=1)+\alpha, \#(x=0)+\beta)}.$$

As you can see, the denominators get cancelled out as hyperparameters remain the same. Recall the Beta function property:

$$B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}.$$

The Beta functions in our expression can hence be simplified to Gamma functions. Additionally, note that,

$$\#(x=1)+\#(x=0) = \#(y=1)+\#(y=0) = N,$$

$$\#(x=1,y)+\#(x=0,y) = \#(y),$$
$$\#(x=1,y)+\#(x=0,y) = \#(x).$$

Therefore, our expression can be simplified to:

$$\frac{\Gamma(\#(x=1,y=0)+\alpha_{1|0})\Gamma(\#(x=0,y=1)+\beta_{1|1})\Gamma(\#(y=0)+\beta)\Gamma(\#(y=1)+\alpha)\Gamma(\#(x=0)+\alpha_{1|0}+\beta_{1|0})\Gamma(\#(x=1)+\alpha_{1|1}+\beta_{1|1})}{\Gamma(\#(y=1,x=0)+\alpha_{1|0})\Gamma(\#(y=0,x=1)+\beta_{1|1})\Gamma(\#(x=0)+\beta)\Gamma(\#(x=1)+\alpha)\Gamma(\#(y=0)+\alpha_{1|0}+\beta_{1|0})\Gamma(\#(y=1)+\alpha_{1|1}+\beta_{1|1})}.$$

(4) The program below is implemented to illustrate that if a table $p(x|y)p(y)$ is more deterministic than table $p(y|x)p(x)$, we should prefer model $M_{x\to y}$. In order to select the more deterministic model we make use of the expression derived for the Bayes Factor in part **(3)**. Bare in mind this expression represents the ratio of the posterior probability of one model $M_{x\to y}$ against that of the other $M_{y\to x}$. We assume uniform priors, setting α, β, $\alpha_{1|1}$, $\alpha_{1|0}$, $\beta_{1|1}$, $\beta_{1|0}$ to one.

```
% GM Assignment 4 -> Question 12.6

clear all;

% situation specified by the question
count_x0_y0 = 10;
count_x0_y1 = 10;
count_x1_y0 = 0;
count_x1_y1 = 0;
count_x0 = 20;
count_x1 = 0;
count_y0 = 10;
count_y1 = 10;

% selected hyperparameter values - uniform priors
alpha10 = 1; alpha11 = 1; alpha = 1;
beta10 = 1; beta11 = 1; beta = 1;

% numerator in (3)
num = gamma(count_x1_y0+alpha10)*gamma(count_x0_y1+beta11)*gamma(count_y1+alpha)*...
          gamma(count_y0+beta)*gamma(count_x0+alpha10+beta10)*...
              gamma(count_x1+alpha11+beta11);

% denominator in (3)
denom=gamma(count_x0_y1+alpha10)*gamma(count_x1_y0+beta11)*gamma(count_x1+alpha)*...
          gamma(count_x0+beta)*gamma(count_y0+alpha10+beta10)*...
              gamma(count_y1+alpha11+beta11);

% bayes factor
BF = num/denom;
```

Running the code gives the following value for the Bayes Factor: 0.1736. This indicates that model $M_{x\to y}$ is more deterministic (and 'preferred')for this numerical example.

Exercise 23.11

A second order HMM is defined as,

$$p(h_{1:T}, v_{1:T}) = p(h_1)p(v_1|h_1)p(h_2|h_1)p(v_2|h_2)\prod_{t=3}^{T} p(h_t|h_{t-1}, h_{t-2})p(v_t|h_t).$$

Note that the problem of computing the most likely joint state $\text{argmax}_{h_{1:T}} p(h_{1:T}|v_{1:T})$ is equivalent to that of computing the most likely path $h_{1:T}$ of $p(h_t|h_{t-1})$. An explicit derivation is obtained by considering,

$$\max_{h_T} \prod_{t=3}^{T} p(h_t|h_{t-1}, h_{t-2})p(v_t|h_t) = \left\{ \prod_{t=3}^{T-1} p(h_t|h_{t-1}, h_{t-2})p(v_t|h_t) \right\} \max_{h_T} p(h_T|h_{T-1}, h_{T-2})p(v_T|h_T),$$

where $\max_{h_T} p(h_T|h_{T-1}, h_{T-2})p(v_T|h_T)$ can be denoted as a message $\mu(h_{T-2})$. Such message transfers information from the end of the chain to the penultimate timestep $T-1$ and also from the end of the chain to the antepenultimate timestep $T-2$. Continuing in this manner, one can define the recursion,

$$\mu(h_{t-2}) = \max_{h_t} p(h_t|h_{t-1}, h_{t-2})p(v_t|h_t)\mu(h_t), \quad 3 \le t \le T,$$

with $\mu(h_T) = 1$. This indicates that maximising over $h_3, ..., h_T$ is condensed into a message $\mu(h_1)$ such that most likely state h_1^* is,

$$h_1^* = \text{argmax}_{h_1} p(h_1)p(v_1|h_1)p(h_2|h_1)p(v_2|h_2)\mu(h_1),$$

Once h_1^* has been calculated, via backtracking for $3 \le t \le T$:

$$h_t^* = \text{argmax}_{h_t} p(h_t|h_{t-1}^*, h_{t-2}^*)p(v_t|h_t)\mu(h_t).$$

Exercise 27.6

The program `demosampleHMMmodified.m` is presented below. This program adjusts `demoSampleHMM.m` to run 100 times for a given λ and a given random draw of the transition and emmision matrices. This procedure is then repeated 20 times for different random draws of the matrices for a given λ. The errors of calculating the smoothed posterior by Gibbs sampling are then averaged over the runs.

```
function demoSampleHMMmodified
% demo of Gibbs sampling from a HMM versus exact result
fprintf(1,['Draw samples from p(h(1:T)|v(1:T)) for a HMM.\nUse these to form ' ...
        'empirical estimates of p(h(t)|v(1:T)) and compare to the exact ' ...
        'p(h(t)|v(1:T))\n\n']);
import brml.*
H=2; V=2; T=10;
% make a HMM
lambda=20;    % specify lambda
runs1 = 20; % repeat procedure 20 times for different random draws of tr/em matrices
runs2 = 100; % runs 100 times for ech transition matrix draw.
n1 = 0; % counter for runs1
tot_mean_abs_error = 0; % keeps track of sum of mean errors to compute avg over runs


while (n1 < runs1)
    A=condp(rand(H,H).^lambda);
    B=condp(rand(V,H));
    a=condp(rand(H,1));
    % draw some samples for v:
    h(1)=randgen(a); v(1)=randgen(B(:,h(1)));
    for t=2:T
        h(t)=randgen(A(:,h(t-1)));  v(t)=randgen(B(:,h(t)));
    end
    [logalpha,loglik]=HMMforward(v,A,a,B); logbeta=HMMbackward(v,A,B);
    gamma=HMMsmooth(logalpha,logbeta,B,A,v); % exact marginal
    n2 = 0; % counter for runs2
    while (n2 < runs2)
    % single site Gibbs updating
        hsamp(:,1)=randgen(1:H,1,T);
        hv=1:T; vv=T+1:2*T; % hidden and visible variable indices
        num_samples=100;
        for sample=2:num_samples
            h = hsamp(:,sample-1);
            emiss=array([vv(1) hv(1)],B);
            trantm=array(hv(1),a);
            trant=array([hv(2) hv(1)],A);
            h(1) = randgen(table(setpot(multpots([trantm trant emiss]), ...
                        [vv(1) hv(2)],[v(1) h(2)]))));
            for t=2:T-1
                trantm.table=A; trantm.variables=[hv(t) hv(t-1)];
                trant.table=A; trant.variables=[hv(t+1) hv(t)];
                emiss.table=B; emiss.variables=[vv(t) hv(t)];
                h(t) = randgen(table(setpot(multpots([trantm trant emiss]),...
                        [vv(t) hv(t-1) hv(t+1)],[v(t) h(t-1) h(t+1)]))));
            end
            trantm.table=A; trantm.variables=[hv(T) hv(T-1)];
            emiss.table=B; emiss.variables=[vv(T) hv(T)];
            h(T) = randgen(table(setpot(multpots([trantm emiss]),[vv(T) hv(T-1)],...
                        [v(T) h(T-1)]))));
            hsamp(:,sample)=h; % take the sample after a forward sweep through time
        end
        for t=1:T
            gamma_samp(:,t) = count(hsamp(t,:),H)/num_samples;
        end
        % update total error
        tot_mean_abs_error = tot_mean_abs_error+ mean(abs(gamma(:)-gamma_samp(:)));
        n2 = n2+1;
    end
    n1 = n1+1;
end
mean_abs_error = tot_mean_abs_error/(runs1*runs2);
fprintf('mean absolute error in the marginal estimate=%g\n', mean_abs_error);
end
```

We obtain mean errors of 0.0347, 0.0423, 0.123 and 0.172 for $\lambda = 0.1, 1, 10, 20$ respectively for 20 runs. The performance of the Gibbs sampling routine is observed to deteriorate with increasing λ. This is due to the following: as λ gets larger, the hidden transition matrix is more deterministic. As a result, we encounter a less random process and the error on the marginal estimate is increased.

Exercise 27.9

(a) The 10 best players for Aces and Bruisers are:

```
Ace_top10 = [9, 11, 2, 10, 12, 16, 19, 20, 1, 7]
Bru_top_10 = [7, 13, 6, 15, 2, 4, 16, 19, 1, 3]
```

The players rating are found by using Gibbs sampling $N = 100000$ times, and repeating $K = 10$ times and find the mean skills rating. K and N can obviously be much larger depending on how patient you are / how much computational power you have. The ratings are found by sampling and calculate probabilities using the given sigmoid function, where we basically take the set of configurations with highest sum of probability associated to the given match outcomes. Code are given below.

(b) Given that we know the Bruisers will field players numbered 1 to 10, the best team configuration for Ace to play is:

```
Ace_plays_best =[11, 12, 2, 14, 4, 13, 19, 16, 20, 10]
```

This is done by sampling $Nb = 100000$ times random configurations of players to play with the opposing teams, and find the team with maximum probability of winning. HOWEVER, if we take the limit of $Nb \to \infty$, the Players that Ace should play must be the same as part (a). This is true regardless of who Bruisers plays. This is intuitive since we are maximizing the chance of winning, which depends on the skill rating differences. Code are also given below.

```
% ==============================================================================

% Setup
load('soccer.mat')

% convert struct data to matrices
AceMatch = zeros(10,20);
BruiserMatch = zeros(10,20);
AceWon = zeros(1,20);

for i = 1:length(game)
```

```
    AceMatch(:,i) = game(i).teamAces;
    BruiserMatch(:,i) = game(i).teamBruisers;
    AceWon(i) = game(i).AcesWin;
end
% ======================================================================

% Part (a)

% repeat the process K times, and we take the mean rating over K trails
K = 10;
Ace_rating = zeros(20,K);
Bru_rating = zeros(20,K);

for j = 1:K

    % Samples N times
    [ Ace_players_best,Bru_players_best,~  ] = samples_27_9gibbs( AceMatch,BruiserMatch,AceWon,100000  ); % SEE BELOW*

    Ace_rating(:,j) = Ace_players_best;
    Bru_rating(:,j) = Bru_players_best;

end

% calculate mean rating of each players across K trails
Mean_Ace_rating = mean(Ace_rating,2);
Mean_Bru_rating = mean(Bru_rating,2);

% Getting the top 10 players
[~,I_A] = sort(Ace_players_best,'descend');
Ace_top10 = I_A(1:10)';

[~,I_B] = sort(Bru_players_best,'descend');
Bru_top_10 = I_B(1:10)';
% ======================================================================

% Part (b)
% now that we have a rating for all players, given Bruisers fielding player 1:10
% we can basically compute the probability of winning the match with certain
% player configurations and simply choose the config that has best chances of
% winning, For Nb -> inf, this should be the same as top10 Ace players in part(a)

% Setup
Bru_players_rating = Mean_Bru_rating(1:10);
idx = linspace(1,20,20);
opt_probability = -inf;

% Optimize for winnign probability, sample randomly N times
Nb = 100000;
for i = 1:Nb
    Ace_plays_sample = datasample(idx,10,'Replace',false); % brute force random sampling
    Ace_players_rating = Mean_Ace_rating(Ace_plays_sample);
    winning_probability = 1/(1+exp(-sum(Ace_players_rating-Bru_players_rating)));

    if winning_probability > opt_probability
        opt_probability = winning_probability;
        Ace_plays_best = Ace_plays_sample;
    end
end
% ======================================================================
```

```
% ======================================================================
% Function for part (a)
function [ Ace_players_best,Bru_players_best,max_score  ] = samples_27_9gibbs( AceMatch,BruiserMatch,AceWon, N )
% ============================================================
% INPUT:
```

```
% AceMatch/BruMatch - players assigned for each match
% AceWon - outcome of the match (+1 if Ace won, -1 otherwise)
% N - number of samples
% ===========================================================
% OUTPUT:
% List of best Ace/Bruiser players from the sampling method, by
% maximizing the 'scores' (sum of probabilities of all games)
% ===========================================================

max_score = -inf;

% initialised randomly, or uniformly, if wanted to, doesnt really matter.
players = unidrnd(5,40,1) - 3; % first 20 is Ace, last 20 is Bruiser
p = 0;

% samples
for it = 1:N

    % Gibbs sampling
    % update one player at a time, while keep others fix
    % (note that if we want to do brute force sampling, we can just
    % assigned a random skills rating to all players at each N, however,
    % neither approach covers the whole spaces of 5^20 combinations)
    if p == 40
        p = 1;
    else
        p = p+1;
    end
    players(p) = unidrnd(5,1,1) - 3;

    Ace_players = players(1:20);
    Bru_players = players(21:40);


    % Calculate total 'scores' for this sets of player ratings
    total_score = 0;
    for match = 1:20
        % get players
        Ace_mi = Ace_players(AceMatch(:,match));
        Bru_mi = Bru_players(BruiserMatch(:,match));
        sigma = 1/(1+exp(-sum(Ace_mi - Bru_mi))); % pbb of Ace wins
        score = ( ((sigma-0.5)*AceWon(match))*10 );
        total_score = total_score + score;
    end

    % Choose the set of rating that have highest 'scores'
    if total_score >= max_score
        max_score = total_score;
        Ace_players_best = Ace_players;
        Bru_players_best = Bru_players;
    end
end
end
% ======================================================================
```

Exercise 27.10

We want to estimate the probability of each disease given symptoms and prior disease probabilities. Specifically, we want to estimate the vector

$$[p(d_1 = 1|\mathbf{s}), \ldots, p(d_D = 1|\mathbf{s})]$$

given the belief network

$$p(s_1, \ldots, s_S, d_1, \ldots, d_D) = \prod_{j=1}^{S} p(s_j|\mathbf{d}) \prod_{i=1}^{D} p(d_i).$$

In general, when using Gibbs Sampling we have to only sample from a conditional distribution $p(x_i|x_{\setminus i})$ at a given time step. This depends only on the Markov blanket of the variable x_i. For a belief network, it is equal to

$$p(x_i|x_{\setminus i}) = \frac{1}{Z} p(x_i|pa(x_i)) \prod_{j \in ch(i)} p(x_j|pa(x_j))$$

where Z is the normalization constant.

In addition, we have to deal with evidence in our case, i.e. the vector of symptoms $\mathbf{s}$. This is easy as we simply clamp the evidential variables to their evidential states and we do not preform sampling from these variables.

In our case at a given time step, we want sample from the distribution

$$p(d_i|\mathbf{d}_{\setminus i}, \mathbf{s})$$

where $\mathbf{d}_{\setminus i}$ is the vector $\mathbf{d}$ with out the element d_i. This is computed as follows

$$\begin{aligned} p(d_i = 0|\mathbf{d}_{\setminus i}, \mathbf{s}) &= \frac{1}{Z} p(d_i = 0) \prod_{j=1}^{S} p(s_j|d_1, \ldots, d_i = 0, \ldots, d_D) & (1) \\ &= \frac{1}{Z} p(d_i = 0) \prod_{j=1}^{S} \sigma(\mathbf{w}_j^T \mathbf{d}_{d_i=0} + b_j) & (2) \end{aligned}$$

where $\mathbf{d}_{d_i=0} = [d_1, \ldots, d_i = 0, \ldots, d_D]$. This is analogously done for $p(d_i = 1|\mathbf{d}_{\setminus i}, \mathbf{s})$ The normalization constant is then

$$Z = p(d_i = 0|\mathbf{d}_{\setminus i}, \mathbf{s}) + p(d_i = 1|\mathbf{d}_{\setminus i}, \mathbf{s})$$

Gibbs sampling produces highly correlated samples so we randomly chose the random variable d_i ordering in which we sample after each round (after we swept through all d_i variables). As recommended in the lecture, we use subsampling of the size equal to the number of random variables the we sample from, i.e. we consider a sample once we have swept through all the random variables d_i. As the initial samples strongly depend on the initial guess, we discard first 1000 samples (burn-in phase). After using subsampling and burn-in, we use 50000 samples in total for the probability estimation.

Finally, the elements of the vector

$$[p(d_1 = 1|\mathbf{s}), \ldots, p(d_D = 1|\mathbf{s})]$$

are estimated as the sample mean of each random variable d_i samples. The numerical values are shown in the code output below.

Code:

1. `ex2710.m`

```
close all ;
clear all ;

run BRMLtoolkit/setup.m ;
import brml.*

load SymptomDisease

% number of diseases
D = numel(p) ;
% number of symptoms
S = numel(s) ;

% sampling parameters
nSamples = 50000 ;
burnIn = 1000 ;

samples = zeros(D, nSamples) ;
% random initialization
samples(:,1) = rand(D, 1) > 0.5 ;
nSample = 2 ;
for nSample=2:nSamples
  ordering = randperm(D) ;
  samples(:, nSample) = samples(:, nSample-1) ;
  if mod(nSample,10000) == 0
    fprintf('n samples: %d\n', nSample) ;
  end
  % sweep through all variables we want to sample from
  for c = 1:D
    % sampling variable d
    di = ordering(c) ;
    % conditioning set
    cSet = setdiff(1:D, di) ;
    % set variables to their evidential state
    d = samples(:,nSample) ;

    % compute conditional p(d_i = 0 |d/d_i, s)
    d(di) = 0 ;
    % p(d_i = 0)
    pd0 = (1-p(di)) ;
    % p(s_j = 1|d)
    ps1 = sigma(W*d + b) ;
    ps0 = 1 - ps1 ;
    psTable = [ps0, ps1] ;
    ps = zeros(S, 1) ;
    for i=1:S
      ps(i) = psTable(i, s(i) + 1) ;
    end
    % p(d_i = 0 |d/d_i, s)
    pdi0Cond = exp(sum(log([pd0; ps]))) ;

    % compute conditional p(d_i = 1 |d/d_i, s)
    d(di) = 1 ;
    % p(d_i = 1)
    pd1 = p(di) ;
    % p(s_j = 1|d)
    ps1 = sigma(W*d + b) ;
    ps0 = 1 - ps1 ;
    psTable = [ps0, ps1] ;
    ps = zeros(S, 1) ;
    for i=1:S
      ps(i) = psTable(i, s(i) + 1) ;
    end
    % p(d_i = 1 |d/d_i, s)
    pdi1Cond = exp(sum(log([pd1; ps]))) ;

    % normalize
    pdiCond = [pdi0Cond; pdi1Cond] ;
    pdiCond = pdiCond / sum(pdiCond) ;

    % sample
    v = pdiCond(1) < rand() ;
    samples(di, nSample) = v ;
  end
end

% burn in and subsample
samples = samples(:, burnIn+1:end) ;
sampleMean = mean(samples, 2)
```

2. `sigma.m`

```
function y = sigma(x)
y = 1 / (1 + exp(-x)) ;
```

Output:

```
0.0030
0.9976
0.0221
1.0000
0.6480
0.0101
0.0146
0.0009
0.0059
0.9999
0.0040
1.0000
1.0000
1.0000
```

```
0.9874
0.9685
0.9876
0.9039
0.9999
0.9943
0.0822
0.7444
0.9999
0.9950
0.0048
0.9994
0.0149
0.0001
0.9987
0.0000
0.0001
0.0909
0.0099
1.0000
0.0000
0.0026
0.9936
0.0011
0.0000
0.0005
0.9929
0.9982
1.0000
0.9998
0.0001
0.0151
0.9968
0.9957
0.0000
0.9997
```

Exercise 27.11

Using marginalisation:

$$p(d|s,D) = \int_\theta p(d|s,\theta)p(\theta|D)d\theta$$

and let $\theta = W, b, p$, we can arrive at

$$p(d|s,D) = \int_{W,b,p} p(d|s,W,b,p)p(W,b,p|D)dWdbdp$$

The latter term can be derived by first considering the Bayes rule:

$$p(W,b,p|D) = \frac{p(W,b,p)p(D|W,b,p)}{p(D)}$$

$$\propto p(W,b,p)p(D|W,b,p)$$

We are given that

$$p(D) = p(s,d) = \prod_j p(s_j|d) \prod_i p(d_i)$$

so

$$p(D|W,b,p) = p(s,d|W,b,p) = \prod_j p(s_j|W,b,p,d) \prod_i p(d_i|W,b,p)$$

now data are given in pairs, symptoms s does not depends on prior probability p, and disease d does not depends in the weight W and bias b. So we have

$$p(D|W,b,p) = \prod_n p(s_n|d_n,W,b)p(d_n|p)$$

and thus we arrived at

$$p(W,b,p|D) \propto p(W,b,p) \prod_n p(s_n|d_n,W,b)p(d_n|p)$$

hence the proof.

In order to estimate $p(d_i = 1|s, D)$, we can simply sample $p(d|s, W, b, p)$ using Gibbs sampling, or other sampling method in a similar manner as the previous question, and compute the integral over W, b and p. This can be done because the $p(W, b, p|D)$ only depends on the set of data points D, which are given (and the normalisation term $p(D)$ can be computed given finite set of data points), and the integral parameters, which can be integrated.