Bayesian Data Analysis, Final

Bugra Gedik
bgedik@cc.gatech.edu

December 6, 2004

Q1)

$$
\begin{aligned}
d|\theta &\sim N(\theta, \sigma^2) \\
\theta|\tau^2 &\sim N(0, \tau^2) \\
p(\tau^2) &\approx (\tau^2)^{-3/4} \\
p(d,\theta,\tau^2) &\approx p(d|\theta)p(\theta|\tau^2)p(\tau^2) \\
p(d,\theta,\tau^2) &\approx N(d|\theta,\sigma^2)N(\theta|0,\tau^2)(\tau^2)^{-3/4} \\
p(\theta,\tau^2|d) &\approx p(d,\theta,\tau^2) \\
p(\theta|d) &\approx \int_0^\infty p(d,\theta,\tau^2)d\tau^2 \\
p(\theta|d) &\approx \int_0^\infty N(d|\theta,\sigma^2)N(\theta|0,\tau^2)(\tau^2)^{-3/4}d\tau^2 \\
p(\theta|d) &\approx e^{-\frac{1}{2\sigma^2}(d-\theta)^2}\int_0^\infty (\tau^2)^{-5/4}e^{-\frac{1}{2\tau^2}\theta^2}d\tau^2 \\
p(\theta|d) &\approx \frac{e^{-\frac{1}{2\sigma^2}(d-\theta)^2}}{\sqrt[4]{\theta^2}} \\
\frac{\partial p(\theta|d)}{\partial\theta} &\approx \frac{e^{-\frac{1}{2\sigma^2}(d-\theta)^2}(2\theta^2-2\theta d+\sigma^2)}{\theta\sqrt[4]{\theta^2}}
\end{aligned}
$$

For $d^2 \geq 2$, the roots of $2\theta^2 - 2\theta d + \sigma^2$ are: $1/2d + 1/2\sqrt{d^2-2\sigma^2}$ and $1/2d - 1/2\sqrt{d^2-2\sigma^2}$. The function $p(\theta|d)$ has limit ∞ around 0. So the modes of the function $p(\theta|d)$ can be summarized as follows: Unimodal with mode 0, if $d^2 < 2$; bimodal with 0 being the first mode and $1/2d + 1/2\sqrt{d^2-2\sigma^2}$ or $1/2d - 1/2\sqrt{d^2-2\sigma^2}$ being the second mode (pick the one resulting in a negative second derivative, which in this case means pick the former if $d > 0$ and the latter if $d < 0$), otherwise. The thresholding function derived from this is as follows:

$$
\delta(\theta|d) = \begin{cases} 0 & d^2 < 2 \\ 1/2d + 1/2\sqrt{d^2-2\sigma^2} & d^2 \geq 2 \text{ and } d > 0 \\ 1/2d - 1/2\sqrt{d^2-2\sigma^2} & d^2 \geq 2 \text{ and } d < 0 \end{cases}
$$

This simplifies to:

$$
\delta(\theta|d) = \begin{cases} 0 & d^2 < 2 \\ \left(1 - \frac{1-\sqrt{1-\frac{2\sigma^2}{d^2}}}{2}\right)d & \text{otherwise} \end{cases}
$$

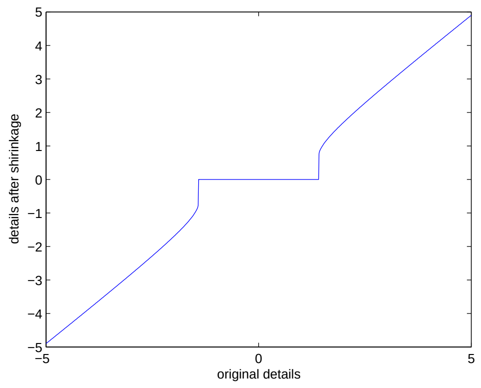

Figure 1: Thresholding function for $\sigma^2 = 1$

Note that,

$$\begin{aligned}\left(1-\frac{1-\sqrt{1-2/d^2}}{2}\right) &\approx \left(1-\frac{1-(1-1/d^2)}{2}\right)d, \text{ for large } d \text{ using } (1-u)^\alpha \approx 1-\alpha u \text{ for small } u \\ &= \left(1-\frac{1}{2d^2}\right)d\end{aligned}$$

I used $\delta(\theta|d)$ for image de-noising. The results and the Matlab code is given below:

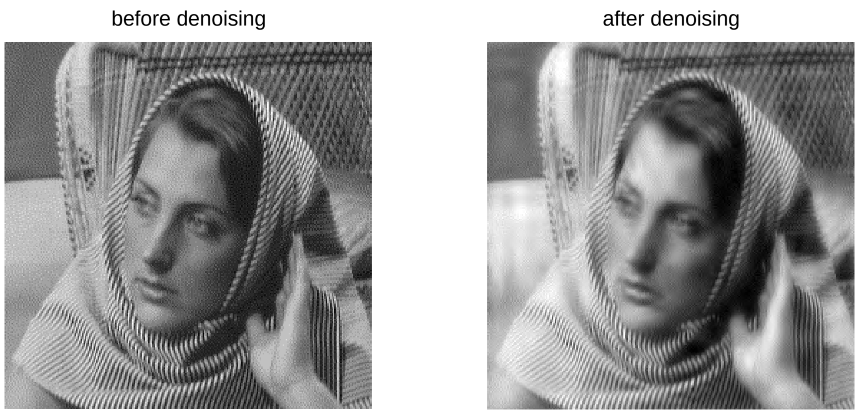

Figure 2: Thresholding function

Q1 Matlab Code

```
clear all;
x = sym('x', 'real'); % used for theta
t = sym('t', 'real'); % used for tau^2
s = sym('s', 'real'); % used for sigma^2
d = sym('d', 'real');


f = 1/(t^1.25) * exp(-(0.5/t)*x^2);
g = simplify( exp(-(0.5/s)*(d-x)^2) * int(f, t, 0, Inf) );
limit(g, x, 0)
pretty(g)
h1 = simplify(diff(g,'x'));
o = solve(h1, 'x');
h2 = diff(h1,'x');
e(1) = subs(h2, 'x', o(1));
e(2) = subs(h2, 'x', o(2));
pretty(o(1)) % output roots
pretty(o(2))
```

```
% use for denoising
figure;
I = load('woman.mat');
I = I.X; % load image
L = min(ceil(log2(size(I))));
I = I(1:2^L,1:2^L); % put into proper scale
I = I / 12.8;
subplot(1,2,1);
imshow(I, []);
title('before denoising');

daub8 = MakeONFilterExt('Daubechies', 16);
V = dwtr_n(I, L, daub8, 0); % transform
D = V(:); % put into vector

v = var(D);
o(1) = subs(o(1), s, v);
o(2) = subs(o(2), s, v);

% apply thresholding
idx = find(D.^2<2);
D(idx) = 0;
idx = find(D.^2>=2 & D>0);
D(idx) = double( subs(o(1), d, D(idx)) );
idx = find(D.^2>=2 & D<0);
D(idx) = double( subs(o(2), d, D(idx)) );

W = reshape(D, size(V)); % back into 2d
W(1) = V(1); % replace smooth
U = idwtr_n(W, L, daub8, 0); % inverse transform
subplot(1,2,2);
imshow(U, []);
title('after denoising');
```

Q2)

a)

$$\begin{aligned}
P(J1, M1|B1) &= \frac{1}{P(B1)} P(B1, J1, M1) \\
&= \frac{1}{P(B1)} \sum_{E,A} P(B1, E, A, J1, M1) \\
&= \frac{1}{P(B1)} \sum_{E,A} P(J1|A)P(M1|A)P(A|B1, E)P(B1)P(E) \\
&= \frac{1}{P(B1)} (\quad P(J1|A0)P(M1|A0)P(A0|B1, E0)P(B1)P(E0) \\
&\quad +P(J1|A0)P(M1|A0)P(A0|B1, E1)P(B1)P(E1) \\
&\quad +P(J1|A1)P(M1|A1)P(A1|B1, E0)P(B1)P(E0) \\
&\quad +P(J1|A1)P(M1|A1)P(A1|B1, E1)P(B1)P(E1) \ ) \\
&= \frac{1}{0.001} (\quad 0.05 * 0.01 * 0.06 * 0.001 * 0.998 \\
&\quad +0.05 * 0.01 * 0.05 * 0.001 * 0.002 \\
&\quad +0.90 * 0.70 * 0.94 * 0.001 * 0.998 \\
&\quad +0.90 * 0.70 * 0.95 * 0.001 * 0.002 \ ) \\
&= 0.5922
\end{aligned}$$

b) From the Matlab BNT Code below, we get: 0.5922
c) From WinBUGS code below (using 100,000 iterations) we get: 0.5913

Q2 Matlab BNT Code

```
function res = q2bnt()

% topologically sorted nodes: B, E, A, J, M
N = 5; % number of nodes
B = 1; E = 2; A = 3; J = 4; M =5; % node numbers

% create the topology, directed acyclic
dag = zeros(N,N); % adjecancy matrix
dag([B,E],A) = 1; dag(A,[J,M]) = 1;

% set domain size for each node
domain_sizes = 2*ones(1,N);

false = 1; true = 2;

% create the bnet
bnet = mk_bnet(dag, domain_sizes, 'names', {'B','E','A','J','M'});
draw_graph(bnet.dag);

% set priors
bnet.CPD{B} = tabular_CPD(bnet, B, [.999, .001]);
bnet.CPD{E} = tabular_CPD(bnet, E, [.998, .002]);

% set conditional probabilities
bnet.CPD{A} = tabular_CPD(bnet, A, [.999, .06, .71, .05, .001, .94, .29, .95]);
bnet.CPD{J} = tabular_CPD(bnet, J, [.95, .10, .05, .90]);
bnet.CPD{M} = tabular_CPD(bnet, M, [.99, .30, .01, .70]);

% set the algorithm
%engine = jtree_inf_engine(bnet);
engine = var_elim_inf_engine(bnet);
%engine = enumerative_inf_engine(bnet);

% set evidence B=true
evidence = cell(1, N);
evidence{B} = true;

% compute probabilities
[engine, loglik] = enter_evidence(engine, evidence);

% find P(J=true,M=true|B=true)
marg = marginal_nodes(engine, [J M]);
res = marg.T(true, true);
```

Q2 WinBUGS Code

```
model
{
  B <-  2;
  E ~  dcat(p.E[]);
  A ~  dcat(p.A[B,E,]);
  J ~  dcat(p.J[A,]);
  M ~  dcat(p.M[A,]);
  Q <-  2*(J-1) + (M-1) + 1;
}

list(
  p.E = c(0.998, 0.002),
  p.A = structure( .Data = c(0.999, 0.001, 0.71, 0.29, 0.06, 0.94, 0.05, 0.95), .Dim = c(2,2,2) ),
  p.J = structure( .Data = c(0.95, 0.05, 0.10, 0.90), .Dim = c(2,2) ),
  p.M = structure( .Data = c(0.99, 0.01, 0.30, 0.70), .Dim = c(2,2) )
)

P{Q = 4} is the interested probability
```

Q3

I used Gibs sampling to solve this problem. 10,000 iterations with burn-in value of 1,000 is used. The resulting histograms representing the posterior distributions of τ, λ, μ are given below. I picked the values of α_λ, β_λ, α_μ, and β_μ as follows: I calculated the means and variances of the first and last 25 values, i.e. $\mathrm{E}[Y_{1:25}]$, $\mathrm{Var}[Y_{1:25}]$, $\mathrm{E}[Y_{end-24:end}]$, and $\mathrm{Var}[Y_{end-24:end}]$. Then I picked α_λ and β_λ such that $\lambda \sim \mathrm{Gamma}(\alpha_\lambda, \beta_\lambda)$ has mean $\mathrm{E}[Y_{1:25}]$ and variance $\mathrm{Var}[Y_{1:25}]$. Similarly $\mu \sim \mathrm{Gamma}(\alpha_\mu, \beta_\mu)$ has mean $\mathrm{E}[Y_{end-24:end}]$ and variance $\mathrm{Var}[Y_{end-24:end}]$.

The corresponding Matlab code is included at the end. It turns out that $\tau = 41$ is the MPE estimate of the change point. This value is quite intuitive based on visual inspection and results in the following partitioning (the first partition ends in 1891). λ has a posterior mean around 3.1, where μ has posterior mean around 0.9.
Partition 1: 4, 5, 4, 1, 0, 4, 3, 4, 0, 6, 3, 3, 4, 0, 2, 6, 3, 3, 5, 4, 5, 3, 1, 4, 4, 1, 5, 5, 3, 4, 2, 5, 2, 2, 3, 4, 2, 1, 3, 2, 2
Partition 2: 1, 1, 1, 1, 3, 0, 0, 1, 0, 1, 1, 0, 0, 3, 1, 0, 3, 2, 2, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 2, 1, 0, 0, 0, 1, 1, 0, 2, 3, 3, 1, 1, 2, 1, 1, 1, 1, 2, 4, 2, 0, 0, 0, 1, 4, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1

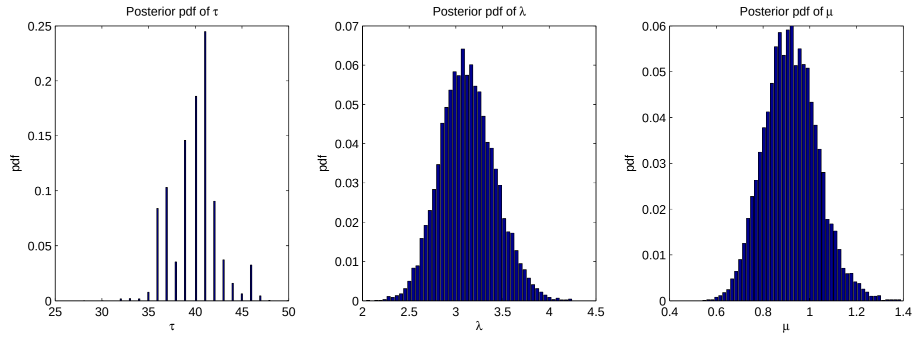

Figure 3: Posterior Probabilities

Q3 Matlab Code

```
l = [1851:1962];   % years
n = length(l); % number of years
% death rates for different years
Y = [ 4,5,4,1,0,4,3,4,0,6,3,3,4,0,2,6,3,3,5,4,5,3,1,4,4,1,5,5,3, ...
      4,2,5,2,2,3,4,2,1,3,2,2,1,1,1,1,3,0,0,1,0,1,1,0,0,3,1,0,3,2, ...
      2,0,1,1,1,0,1,0,1,0,0,0,2,1,0,0,0,1,1,0,2,3,3,1,1,2,1,1,1,1, ...
      2,4,2,0,0,0,1,4,0,0,0,1,0,0,0,0,0,1,0,0,1,0,1 ];

K = 25;
YM1 = mean(Y(1:K));
YV1 =  var(Y(1:K));
YM2 = mean(Y(end-K+1:end));
YV2 =  var(Y(end-K+1:end));

% hyper parameters
beta_lambda  = YM1 / YV1;
alpha_lambda = YM1 * beta_lambda ;
beta_mu      = YM2 / YV2;
alpha_mu     = YM2 * beta_mu ;

% initialize parameters
lambda = rand_gamma(alpha_lambda, beta_lambda, 1, 1);
mu     = rand_gamma(alpha_mu, beta_mu, 1, 1);
tau    = ceil ( rand * n );

% gibs parameters
```

```
itrcnt = 10000; burnin = 1000;
effcnt = itrcnt - burnin;

% collected parameter values
thetas = zeros(effcnt, 3);

% perform gibs
for iter=1:itrcnt,
    % tau step
    for k=1:n,
        ps(k) = exp( (mu-lambda)*k + log(lambda/mu)*sum(Y(1:k)) );
    end
    ps = ps / sum(ps);
    r = rand;
    for k=1:n,
        r = r - ps(k);
        if(r<=0) break; end
    end
    tau = k;

    % lambda step
    ga = alpha_lambda + sum(Y(1:tau)); % mean
    gb = beta_lambda + tau; % variance
    lambda = rand_gamma(ga, gb);

    % mu step
    ga = alpha_mu + sum(Y(tau+1:n)); % mean
    gb = beta_mu + (n - tau); % variance
    mu = rand_gamma(ga, gb);

    % collect
    eiter = iter - burnin;
    if(eiter > 0)
        thetas(eiter, :) = [tau lambda mu];
    end
end

subplot(1,3,1);
taus = thetas(:, 1);
[cnt, pos] = hist(taus, n);
bar(pos, cnt/sum(cnt));
title('Posterior pdf of \tau');
xlabel('\tau'); ylabel('pdf');

B = 50;
subplot(1,3,2);
lambdas = thetas(:, 2);
[cnt, pos] = hist(lambdas, B);
bar(pos, cnt/sum(cnt));
title('Posterior pdf of \lambda');
xlabel('\lambda'); ylabel('pdf');

subplot(1,3,3);
mus = thetas(:, 3);
[cnt, pos] = hist(lambdas, B);
bar(pos, cnt/sum(cnt));
title('Posterior pdf of \mu');
```

```
xlabel('\mu'); ylabel('pdf');
```