Final Exam

Ni Wang

1 Bayesian Wavelet Shrinkage

Figure 1 shows the famous blocky function from Donoho and Johnstone (1994). Figure 2 shows the plot of noisy function when random noise is added to the function.

The signal is contaminated with additive noise which is distributed evenly among all wavelet coefficients. The support of the wavelet falls entirely within a portion of the blocky function that is completely flat. So, there are many coefficients at the higher levels that are exactly zero. When noise is added, the resulting "zero" coefficients will become normal random variables.

Figure 3 and Figure 4 shows the recovered signal from contaminated noisy signal. We may see that the hard thresholding performs much better than the linear Bayes shrinkage when the original signal has blocky characteristics.

2 Alarm

(1). Calculate the exact probability

$$P(J1, M1|B1) = \frac{1}{P(B1)} P(B1, J1, M1) = \frac{1}{P(B1)} \sum_{E,A} P(B1, E, A, J1, M1)$$

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

(2). Use Kevin Murphy's BNT toolkit, following matlab code is used to calculate $P(J1, M1|B1)$.

Matlab Source Code:

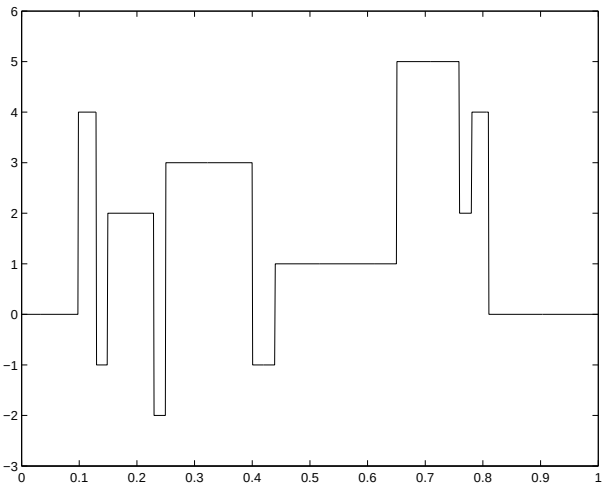

Figure 1: The blocky function of Donoho and Johnstone (1994)

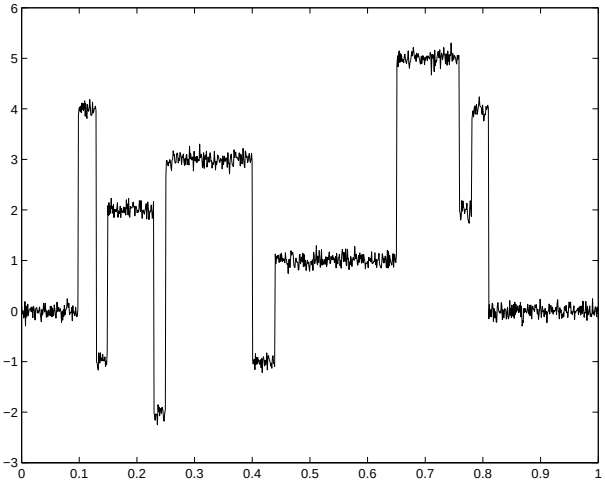

Figure 2: The blocky function with added noise

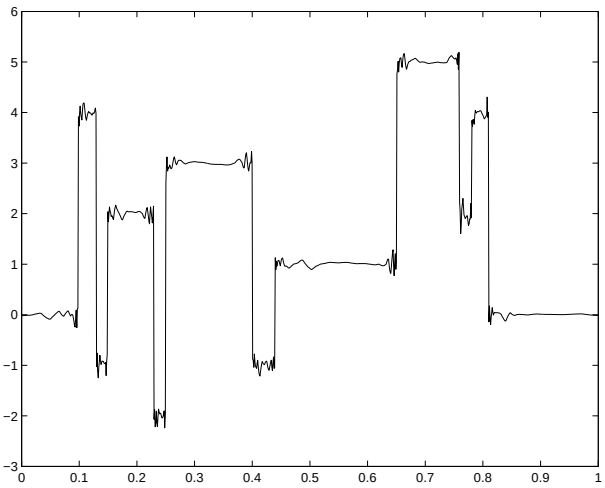

Figure 3: The recovered blocky function using hard thresholding

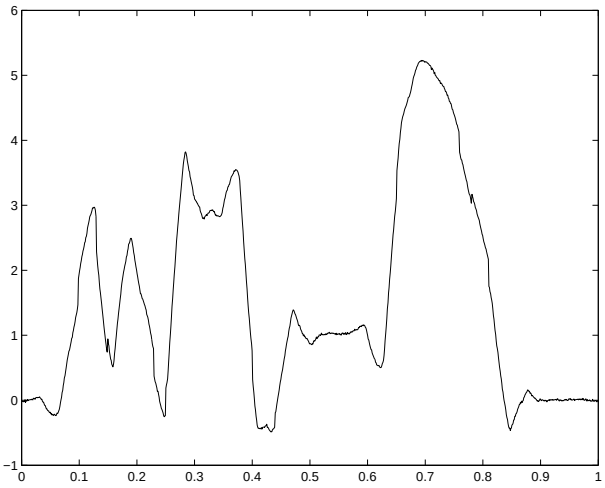

Figure 4: The recovered blocky function using linear Bayes shrinkage method

```
N = 5;
dag = zeros(N,N);
B = 1; E = 2; A= 3; J = 4; M = 5;
dag([B,E],A) = 1;
dag(A,[J,M]) = 1;
node_sizes = 2*ones(1,N);
bnet = mk_bnet(dag, node_sizes, 'names', {'B','E','A','J','M'});
draw_graph(bnet.dag);
false = 1; true = 2;
bnet.CPD{B} = tabular_CPD(bnet, B, [.999, .001]);
bnet.CPD{E} = tabular_CPD(bnet, E, [.998, .002]);
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]); engine =
jtree_inf_engine(bnet);
evidence = cell(1,N);
evidence{B} = true;
[engine, loglik] = enter_evidence(engine, evidence);
marg = marginal_nodes(engine, M);
p1=marg.T(true)
evidence{M}=true;
[engine, loglik] = enter_evidence(engine, evidence);
marg = marginal_nodes(engine, J);
p2=marg.T(true);
p=p1*p2
```

The result is given as follows:

```
p1 = 0.6586
p2 = 0.8992
p  = 0.5922
```

where $p1 = P(M1|B1)$, $p2 = P(J1|M1, B1)$ and $p = P(J1, M1|B1)$.

(3). Using the BUGS.

First, we calculate $P(M1|B1)$, the BUGS code is as follows:

```
model Alert;
```

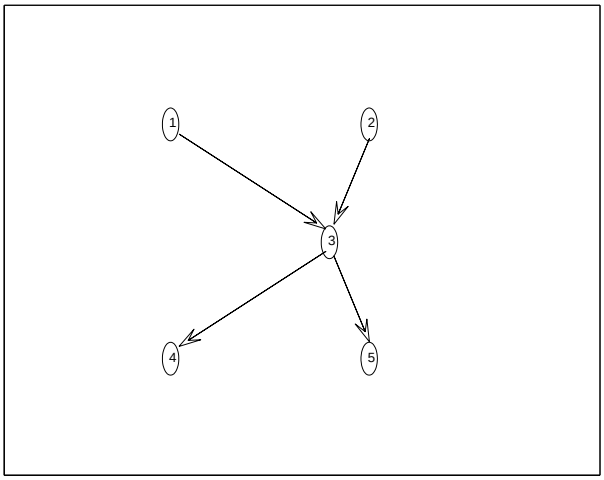

Figure 5: The DAG output of the Matlab file

```
{  burglary ~dcat(p.burglary[]);
   earthquake  ~ dcat(p.earthquake[]);
   alarm ~ dcat(p.alarm[burglary, earthquake, ])
   john ~ dcat(p.john[alarm,]);
   mary  ~ dcat(p.mary[alarm,]);
}
list(
        p.earthquake=c(0.998, 0.002),
        p.alarm = 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.john = structure(.Data = c(0.95,0.05,0.10,0.90), .Dim = c(2,2)),
        p.mary = structure(.Data = c(0.99,0.01,0.30,0.70), .Dim = c(2,2)),
        burglary = 2
   )
list( earthquake = 1, alarm = 1, john = 1, mary = 1)
```

After 10000 iterations, we obtain the mean value of M as $\mathrm{E}M = 2 * p_M + (1 - p_M) = 1.661$, that is, $P(M1|B1) = p_M = 1.661 - 1 = 0.661$. This answer is similar to the $p1 = 0.6586$ obtained using BNT method.

Next, we change the data by setting $B = M = 2$

```
 list(
```

```
        p.earthquake=c(0.998, 0.002),
        p.alarm = 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.mary = structure(.Data = c(0.99,0.01,0.30,0.70), .Dim = c(2,2)),
        burglary = 2,
        mary=2
  )
```

and change the initial values to

```
list(earthquake = 1, alarm = 1, john = 1)
```

After 10000 iterations, the mean value of J is obtained as $\mathrm{E}J = 2 * p_J + (1 - p_J) = 1.899$. Then, $P(J1|M1, B1) = p_J = 0.899$. Thus, the final result $P(J1, M1|B1) = P(J1|M1, B1)P(M1|B1) = p_J p_M = 0.899 * 0.661 = 0.5942 \approx 0.5922$ as in the exact probability calculation.

3 Change Point Analysis

First, we specify the joint distribution as follows:

$$\begin{aligned} f(y, \tau, \lambda, \mu) &\propto \prod_{i=1}^{\tau} f(y_i|\lambda) \prod_{i=\tau+1}^{n} f(y_i|\mu)\pi(\tau)\pi(\lambda)\pi(\mu) \\ &\propto \frac{\lambda^{(\sum_{i=1}^{\tau} y_i)+\alpha_\lambda-1}\mu^{(\sum_{i=1+\tau}^{n} y_i)+\alpha_\mu-1}}{\prod_{i=1}^{n} y_i!} \exp\{-\lambda(\tau+\beta_\lambda)\}\exp\{-\mu(n-\tau+\beta_\lambda)\} \end{aligned}$$

To find the full conditional for τ, we select the terms from $f(y, \mu, \tau, \lambda)$ that contain τ and normalize. Since, $\pi(\tau|\mu, \lambda, y) \propto f(y, \mu, \tau, \lambda)$. We have,

$$p(\tau = k|\mu, \lambda, Y) \propto \lambda^{\alpha_\lambda-1+\sum_{i=1}^{k} Y_i} \times \exp\{-(k+\beta_\lambda)\lambda\} \times \mu^{(\sum_{i=1+k}^{n} y_i)+\alpha_\mu-1} \times \exp\{-\mu(n-k+\beta_\lambda)\}$$

Similarly, we have full conditional probabilities for λ and μ as follows:

$$\begin{aligned} \pi(\lambda|\tau, \mu, Y) &\propto \lambda^{(\sum_{i=1}^{\tau} y_i)+\alpha_\lambda-1}\exp\{-\lambda(\tau+\beta_\lambda)\} \\ \pi(\mu|\tau, \lambda, Y) &\propto \mu^{(\sum_{i=\tau+1}^{n} y_i)+\alpha_\mu-1}\exp\{-\mu(n-\tau+\beta_\mu)\} \end{aligned}$$

That is, $[\lambda|\tau, \mu, Y] \sim Gamma(\alpha_\lambda + \sum_{i=1}^{\tau} Y_i, \beta_\lambda + \tau)$ and
$[\mu|\tau, \lambda, Y] \sim Gamma(\alpha_\mu + \sum_{i=1+\tau}^{n} Y_i, \beta_\mu + n - \tau)$.

	mean	median	std
λ	3.1133	3.1023	0.2898
μ	0.9250	0.9207	0.1156
τ	39.9821	40	2.4584

Table 1: The MCMC output for the model parameters

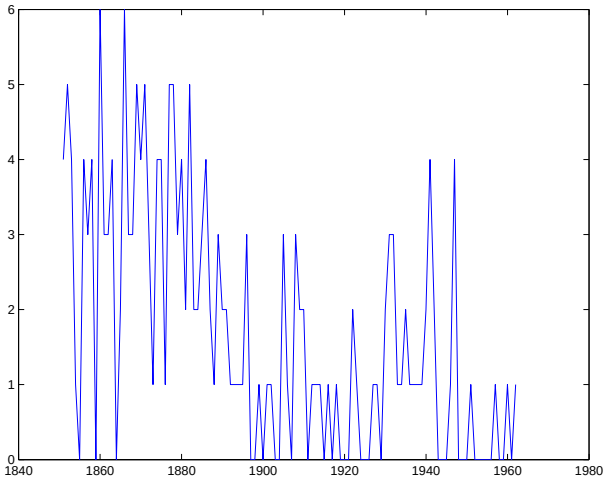

Figure 6: The plot of the original data

Then, τ is generated according to a discrete distribution with

$$P(\tau = k|\mu, \lambda, Y) = \frac{e^{-(\lambda-\mu)k} \times (\lambda/\mu)^{\sum_{i=1}^{k} Y_i}}{\sum_{k=1}^{n}\{e^{-(\lambda-\mu)k} \times (\lambda/\mu)^{\sum_{i=1}^{k} Y_i}\}}$$

We run 10,000 simulations (Gibbs samplers with burn-in 1000). The summary statistics for λ, μ, τ are given in the Table (1). Figure 3, 3 below show the corresponding MCMC traces and histograms obtained by the Gibbs sampler. The Matlab code is attached behind.

The Matlab code for Gibbs sampler is as follows:

```
% This script constructs Gibbs samplers to find the Bayes estimator for
% the change point problem in final 3
close all; clear all;


tt=[1851:1962];
minedata = [
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,...
```

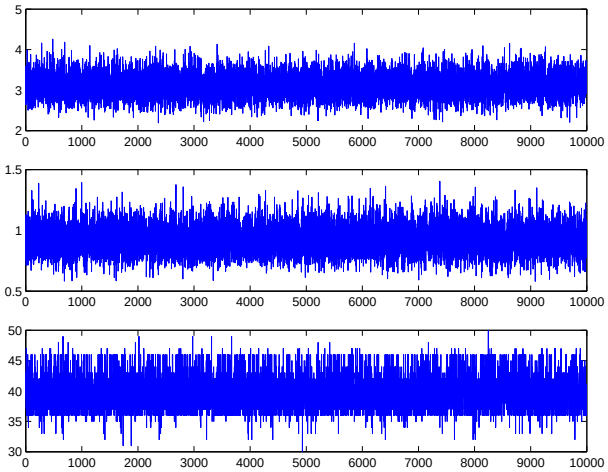

Figure 7: The MCMC trace for λ, μ, and τ

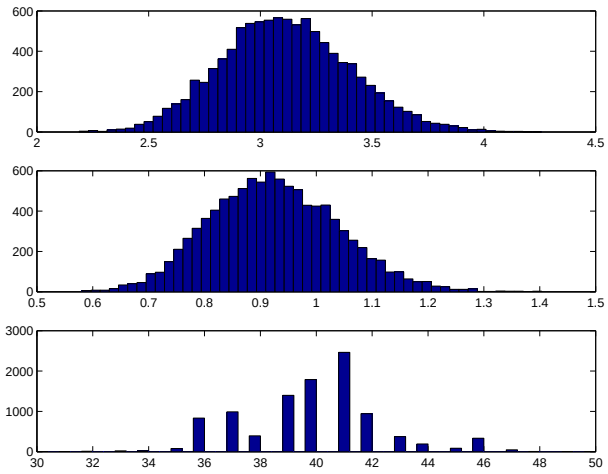

Figure 8: The histograms for λ, μ, and τ

```
                    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];
y=minedata'; n=length(minedata);
% figure(1);
% plot(tt, y);
% print -depsc 'c:\temp\plot.eps'
% set the hyper priors
alpha_lambda=3; beta_lambda=1; alpha_mu=1;  beta_mu=1; iterations
= 10000; burn=1000;
%-------------------------
% initial parameters
tau=40; lambda=5; mu=3;
p=zeros(1,n); % the initial cell probability for tau
indicator=zeros(1,n);
%--------------------------
% for iter=1:iterations
mus=[]; lambdas=[]; taus=[]; h=waitbar(0,'simulation in process');
for iter = 1 : burn + iterations
    indicator=zeros(1,n);
    indicator(1:tau)=1;
    new_lambda= gamrnd(alpha_lambda+ indicator*y , 1/(beta_lambda+tau) );
    new_mu=gamrnd(alpha_mu + sum(y)-indicator*y, 1/(beta_mu + n -tau)  );

    % calculate the cell probability for tau
    p=zeros(1,n);
    q=zeros(1,n);
    for i=1:n
        indicator=zeros(1,n);
        indicator(1:i)=1;
        p(i)= exp(-(lambda-mu)* i ) * (lambda/mu)^(indicator*y);
    end
    p=p ./ sum(p); % normalize
    % generate tau according to the discrete distribution
    u=rand; q=0;

    i=1;
```

```
    while (u > q)
         q=q+p(i);
         i=i+1;
    end
    new_tau=i-1;

    lambdas=[lambdas new_lambda];
    mus= [mus new_mu];
    taus=[taus new_tau];
    mu=new_mu;
    tau=new_tau;
    lambda=new_lambda;
    waitbar(iter/(burn+iterations))
end

close(h);

lambdass=lambdas(burn+1:iterations+burn); mlambda=mean(lambdass)
median_lambda=median(lambdass) sd_lambda=std(lambdass)
muss=mus(burn+1:iterations+burn); mmu=mean(muss)
median_mu=median(muss) sd_mu=std(muss)
tauss=taus(burn+1:iterations+burn); mtau=mean(tauss)
median_tau=median(tauss) sd_tau=std(tauss)

figure(2) subplot(3,1,1) plot(lambdass) subplot(3,1,2) plot(muss)
subplot(3,1,3) plot(tauss) print -depsc 'c:\temp\gibbs1.eps'

figure(3) subplot(3,1,1) hist(lambdass, 50) subplot(3,1,2)
hist(muss, 50) subplot(3,1,3) hist(tauss, 50) print -depsc
'c:\temp\gibbs2.eps'
```