ECE5412 Yucheng Chen NetID: 3075503

Problem 49.

Consider

Therefore,

Consider is positive definite, then we have

So, for any function

Problem 51

Problem 54

The transition probability is:

If there exists a stationary distribution such that

Then, we have:

Consider is symmetric, .

Then we have:

The equation above applies to any . Therefore, the stationary distribution is proportional to .

Problem 55

I chose a dataset which records the Australian Credit Approval. The dataset has 14 features for each data point along with one classification label. I implemented LDA and logistic regression for the dataset. The correct classification rate for LAD is 92.75% and for logistic regression is 88.41%. The performance of two methods are close but the logistic regression needs more training time.

The MATLAB code is attached below:

%problem 55

clear,clc,close all

data = importdata('australian');

data = data.textdata;

[numData, numFeature] = size(data);

dataset = zeros(numData,numFeature);

for i = 1:numData

for j = 1:numFeature

cell\_data = str2num(data{i,j});

dataset(i,j) = cell\_data(1);

end

end

randIndices = randperm(numData);

trainingNum = round(numData\*0.9);

trainingData = dataset(randIndices(1:trainingNum),:);

testData = dataset(randIndices(trainingNum+1:end),:);

MdLinear = fitcdiscr(trainingData(:,2:end),trainingData(:,1));

predictedClass\_LDA = predict(MdLinear, testData(:,2:end));

predictedClass\_LDA = max(predictedClass\_LDA,0);

x0 = rand(1,13)/100;

trainingFeature = trainingData(:,2:end);

trainingLabel = max(trainingData(:,1),0);

options = optimset('PlotFcns',@optimplotfval,'MaxIter',100000,'MaxFunEvals',100000);

[x,val,etflag] = fminunc(@(x)costFcn(x,trainingFeature,trainingLabel),x0,options);

predicted\_LR = 1./(1+exp(-testData(:,2:end)\*x'));

for i = 1:length(predicted\_LR)

if predicted\_LR(i) >= 0.5

predicted\_LR(i) = 1;

else

predicted\_LR(i) = 0;

end

end

testLabel = max(testData(:,1),0);

LDA\_rate= 0;

LR\_rate = 0;

for i = 1:length(testLabel)

if abs(predicted\_LR(i)-testLabel(i)) < 0.01

LR\_rate = LR\_rate + 1;

end

if abs(predictedClass\_LDA(i) - testLabel(i)) < 0.01

LDA\_rate = LDA\_rate + 1;

end

end

function cost = costFcn(x,trainingData,trainingLabel)

cost = 0;

numData = size(trainingData,1);

for i = 1:numData

dataline = trainingData(i,:);

sumLine = sum(dataline.\*x);

y = trainingLabel(i);

p = 1/(1+exp(-sumLine));

cost = cost - (y\*log(p) + (1-y)\*log(1-p));

end

end

Problem 57 & 58

I use real world example that uses both MCMC and Gibbs sampling.

The Ricker model is one classical discrete population model, which gives the expected number of individuals in generation as a function of the number of individuals in the previous generation . This model is described by the following equation:

where is the maximum per capita growth rate, is the environmental carrying capacity. The log-transformation of Ricker model is written as

where

The problem is that assume we know the prior distribution of , the values of and a set of observations , how can we estimate the posterior of ?

where

The objective is to sample from the distribution . However, it is very hard to analytically compute this posterior. Then we can use Gibbs sampling: we first sample , then we sample .Then the collection of formulates the posterior of .

The generate steps of the algorithm are as follows:

*Step 1: Define a prior of parameter and obtain a sample ;*

*Step 2:*

*Loop://Gibbs sampling*

*//MH sampling*

*until meet some terminating conditions*

Then we look at the details of two sampling step inside the Gibbs sampling loop:

1. We can use Sequential Monte Carlo (SMC) to sampling from .

Step 1:

Step 2: for each

for

Sample according to the corresponding

1. Use Metropolis-Hastings (MH) algorithm to sample from

Sample from prior

Sample

Loop :

If

else

If

else

Uniformly sample from , we obtain

In the implementation, I set the known parameters to be

Generate a set of observations with and .

Then, I run the algorithm and the posterior of the parameter is shown as the following plot:



As we can observe, the peak of the distribution is very closed to the ground truth value The mean of all samples is 0.9418, which is closed to 1 and the variance of the posterior is 8.0818, which is significantly reduced compared to the prior.

MATLAB code:

%p57,p58

clc,clear,close all

%generate observations Y

a = 1;

b = 0.5;

omega\_v = .5;

omega\_n = 1;

x0 = normrnd(0,1,1);

T = 40;

Y = zeros(1,T);

x\_cur = x0;

for i = 1:T

x\_cur = x\_cur + a - b\*exp(x\_cur) + normrnd(0,omega\_v);

y = x\_cur + normrnd(0,omega\_n);

Y(i) = y;

end

%estimate the parameters a, b

ab = mvnrnd([0,0],[100,0;0,2]);

[X,x0] = sampleX(ab(1),b,T,omega\_v);

N = 1000; %gibbs sampling iterations

%Gibbs sampling loop

As = zeros(1,N);

Bs = zeros(1,N);

for i = 1:N

paramSamples = MCMC\_params(X,Y,omega\_v,omega\_n,x0);

param = paramSamples(:,unidrnd(size(paramSamples,2)))

[X,x0] = SMCTraj(param(1),b,omega\_v,omega\_n,Y,T);

As(i) = param(1);

Bs(i) = param(2);

end

function [X,x0] = SMCTraj(a,b,omega\_v,omega\_n,Y,T)

N = 8000;

x0 = normrnd(0,1,[N,1]);

sampleTrajs = zeros(N,T+2);

sampleTrajs(:,2) = x0;

for j = 1:N

x\_cur = x0(j);

for i = 1:T

x\_cur = x\_cur + a - b\*exp(x\_cur) + normrnd(0,omega\_v);

sampleTrajs(j,i+2) = x\_cur;

y = Y(i);

sampleTrajs(j,1) = sampleTrajs(j,1) + log(max(normpdf(y,x\_cur,omega\_n),exp(-100)));

end

end

[~,sampleTrajIndex] = max(sampleTrajs(:,1));

x0 = sampleTrajs(sampleTrajIndex,2);

X = sampleTrajs(sampleTrajIndex,3:end);

end

function [X,x0] = sampleX(a,b,T,omega\_v)

X = zeros(1,T);

x0 = normrnd(0,1,1);

x\_cur = x0;

for i = 1:T

x\_cur = x\_cur + a - b\*exp(x\_cur) + normrnd(0,omega\_v);

X(i) = x\_cur;

end

end

function params = MCMC\_params(X,Y,omega\_v,omega\_n,x0)

ab0 = mvnrnd([0,0],[2,0;0,2]);

N = 8000;

%a = a0;b = b0;

a\_prev = ab0(1);b\_prev = 0.5;%b\_prev = ab0(2);

params = zeros(2,N);

%params(:,1) = [a0;b0];

for i = 1:N

ab = mvnrnd([a\_prev,b\_prev],[0.005,0;0,0.005]);%proposal distribution

a = ab(1);

%b = ab(2);

%%%%

b = 0.5;

%%%

logProb1 = evalTrajectoryLogProb(a\_prev,b\_prev,omega\_v,omega\_n,X,Y,x0);

logProb2 = evalTrajectoryLogProb(a,b,omega\_v,omega\_n,X,Y,x0);

logProb1 = logProb1 + normpdf(a,a\_prev,0.005);

logProb2 = logProb2 + normpdf(a\_prev,a,0.005);

if logProb2 > logProb1

params(:,i) = [a,b];

a\_prev = a;

b\_prev = b;

else

u = rand();

if u <= exp(logProb2-logProb1)

params(:,i) = [a;b];

a\_prev = a;

b\_prev = b;

else

params(:,i) = [a\_prev;b\_prev];

end

end

end

end

function logProb = evalTrajectoryLogProb(a,b,omega\_v,omega\_n,X,Y,x0)

logProb = log(normpdf(x0,3,1));

x\_prev = x0;

for i = 1:length(X)

x = X(i);

y = Y(i);

logProb = logProb + log(max(normpdf(x,x\_prev + a - b\*exp(x\_prev),omega\_v),exp(-100))) + log(max(normpdf(y,x,omega\_n),exp(-100)));

end

end

Reference:

Gao, M., Chang, X., & Wang, X. (2012). Bayesian parameter estimation in dynamic population model via particle Markov chain Monte Carlo.

Problem 59

Suppose is an one-dimensional stochastic process. In the simulation, I set and the process noise

Given the sample set at time step :

For

endFor

The Distribution of at







MATLAB code:

clc,clear,close all

x0 = normrnd(0,1000,[1,1000]);

T = 1000;

omega\_var = 2;

x = x0;

x\_next = [];

for i = 1:T

for j = 1:length(x)

x\_next = [x\_next,normrnd(sin(x(j)),omega\_var)];

end

x = x\_next;

hist(x,50)

end

Problem 61.

There are 4 state variables. After applying Kalman filter to the problem , we can obtain the time history of real state variables and estimated state variables as follows:





The mean absolute errors of the four state estimations are Also, given that the acceleration is constant, the trajectories of velocities are lines and the trajectories of positions are parabola.

The MATLAB code is shown as follows:

clc,clear,close all

T = 0.1;

A = [1 T 0 0;0 1 0 0;0 0 1 T;0 0 0 1];

C = [1 0 0 0;0 0 1 0];

f = [T^2/2 0;T 0;0 T^2/2;0 T];

r = [1;2];

z0 = [0;0.1;0;-0.1];

z = z0;

sigma\_ob = 1;

sigma\_process = 0.1;

Q = diag(sigma\_process^2\*[1,1,1,1]);

R = diag(sigma\_ob^2\*[1,1]);

P = eye(4);

z\_est0 = zeros(4,1);

z\_est = z\_est0;

z\_array = [];

z\_est\_array = [];

tspan = 10000;

for i = 1:tspan

omega = normrnd(0,sigma\_process,[4,1]);

niu = normrnd(0,sigma\_ob,[2,1]);

z = A\*z + f\*r + omega;

y = C\*z + niu;

z\_est = A\*z\_est + f\*r;

P = A\*P\*A'+Q;

K = P\*C'\*inv(C\*P\*C' + R);

z\_est = z\_est + K\*(y-C\*z\_est);

P = (eye(4) - K\*C)\*P;

z\_array = [z\_array,z];

z\_est\_array = [z\_est\_array,z\_est];

end

figure(1)

plot(T\*(1:tspan),z\_array(1,:),T\*(1:tspan),z\_est\_array(1,:))

legend('z(1)','z\_{est}(1)')

figure(2)

plot(T\*(1:tspan),z\_array(2,:),T\*(1:tspan),z\_est\_array(2,:))

legend('z(2)','z\_{est}(2)')

figure(3)

plot(T\*(1:tspan),z\_array(3,:),T\*(1:tspan),z\_est\_array(3,:))

legend('z(3)','z\_{est}(3)')

figure(4)

plot(T\*(1:tspan),z\_array(4,:),T\*(1:tspan),z\_est\_array(4,:))

legend('z(4)','z\_{est}(4)')

mean(abs(z\_array(1,:)-z\_est\_array(1,:)))

mean(abs(z\_array(2,:)-z\_est\_array(2,:)))

mean(abs(z\_array(3,:)-z\_est\_array(3,:)))

mean(abs(z\_array(4,:)-z\_est\_array(4,:)))

Problem 62.

(a). The MATLAB code is attached as follows:

clc,clear,close all

tspan = 1000;

S = [0, 10, 20];

P = [0.1,0.1,0.8;0.3,0.3,0.4;0.2,0.2,0.6];

b = [0.3,0.3,0.4];

sigma\_sq = 5;

pi\_s = b;

p\_error = 0;

for i =1:tspan

b = b\*P;

x = 10\*(find(mnrnd(1,b))-1);

y = normrnd(x,sqrt(sigma\_sq));

pi\_s\_unnorm = [normpdf(y,0,sqrt(sigma\_sq))\*pi\_s\*P(:,1),...

normpdf(y,10,sqrt(sigma\_sq))\*pi\_s\*P(:,2),...

normpdf(y,20,sqrt(sigma\_sq))\*pi\_s\*P(:,3)];

pi\_s = pi\_s\_unnorm/sum(pi\_s\_unnorm);

p\_error = 1 - max(pi\_s) + p\_error;

end

ave\_p\_error = p\_error/tspan

The average probability error corresponding to difference measurement noise are listed as follows:s

(b). The MATLAB code is shown as follows:

clc,clear,close all

tspan = 1000;

S = [0, 10, 20];

P = [0.1,0.1,0.8;0.3,0.3,0.4;0.2,0.2,0.6];

b0 = [0.3,0.3,0.4];

sigma\_sq = 1;

pi\_s = b0;

b = b0;

p\_error = 0;

ys = [];

for i =1:tspan

b = b\*P;

x = 10\*(find(mnrnd(1,b))-1);

y = normrnd(x,sqrt(sigma\_sq));

ys = [ys,y];

pi\_s\_unnorm = [normpdf(y,0,sqrt(sigma\_sq))\*pi\_s\*P(:,1),...

normpdf(y,10,sqrt(sigma\_sq))\*pi\_s\*P(:,2),...

normpdf(y,20,sqrt(sigma\_sq))\*pi\_s\*P(:,3)];

pi\_s = pi\_s\_unnorm/sum(pi\_s\_unnorm);

p\_error = 1 - max(pi\_s) + p\_error;

end

p\_error\_tot = 0;

for i = 1:tspan

%forward

pi\_s = b0;

for j = 1:i

pi\_s\_unnorm = [normpdf(ys(j),0,sqrt(sigma\_sq))\*pi\_s\*P(:,1),...

normpdf(ys(j),10,sqrt(sigma\_sq))\*pi\_s\*P(:,2),...

normpdf(ys(j),20,sqrt(sigma\_sq))\*pi\_s\*P(:,3)];

pi\_s = pi\_s\_unnorm/sum(pi\_s\_unnorm);

end

%backward

beta = [1,1,1]';

for k = tspan:-1:i

beta = P\*diag([normpdf(ys(k),0,sqrt(sigma\_sq)),normpdf(ys(k),10,sqrt(sigma\_sq)),normpdf(ys(k),20,sqrt(sigma\_sq))])\*beta;

beta = beta/sum(beta);

end

pi\_tot = [pi\_s(1)\*beta(1),pi\_s(2)\*beta(2),pi\_s(3)\*beta(3)];

pi\_tot = pi\_tot/sum(pi\_tot);

p\_error\_tot = 1 - max(pi\_tot) + p\_error\_tot;

end

ave\_p\_error = p\_error/tspan

ave\_p\_error\_tot = p\_error\_tot/tspan

As we can observe that, as the measurement noise increases, the average probability error increases. Also, optimal smoother has less average probability error than optimal filter given a measurement noises.