CPSC-540 Machine Learning 2007

Homework # 3

Due Thursday March 29 in class if you would like feedback before exam. Otherwise Tuesday April 3.

NAME:_____

Signature:_____

STD. NUM: _____

General guidelines for homeworks:

You are encouraged to discuss the problems with others in the class, but all write-ups are to be done on your own.
Homework grades will be based not only on getting the "correct answer," but also on good writing style and clear presentation of your solution. It is your responsibility to make sure that the graders can easily follow your line of reasoning.
Try every problem. Even if you can't solve the problem, you will receive partial credit for explaining why you got stuck on a promising line of attack. More importantly, you will get valuable feedback that will help you learn the material.
Please acknowledge the people with whom you discussed the problems and what sources you used to help you solve the problem (e.g. books from the library). This won't affect your grade but is important as academic honesty.
When dealing with Matlab exercises, please attach a printout with all your code and show your results clearly.

1. **Importance sampling for Logistic Regression**:

   The file `islogit.m`, consists of input-output i.i.d. data sets $x \triangleq x_{1:T} \triangleq \{x_0, x_1, \ldots, x_T\}$ and $y \triangleq y_{1:T} \triangleq \{y_0, y_1, \ldots, y_T\}$, where $x_t \in \mathbb{R}$ and $y_t \in \{0, 1\}$. The idea is to come up with a model that takes a new input x_{T+1} and produces as output $p(y_{T+1} = 1|x_{T+1})$ and $p(y_{T+1} = 0|x_{T+1})$. This classification problem arises in several areas of technology, including condition monitoring and binary decision systems. For example, when monitoring patients, we might wish to decide whether they require an increase in drug intake based on new evidence.

   For practical reasons, we parameterise our model. In particular, we introduce the following Bernoulli likelihood function:

$$p(y_t|x_t, \theta) = \left[\frac{1}{1 + \exp(-\theta x_t)}\right]^{y_t} \left[1 - \frac{1}{1 + \exp(-\theta x_t)}\right]^{1-y_t}$$

   where θ are the model parameters. The logistic function $p(y_t = 1|x_t) = \frac{1}{1+\exp(-\theta x_t)}$ is conviniently bounded between 0 and 1.

   We also assume a Gaussian prior

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(\theta - \mu)'(\theta - \mu)\right)$$

   The goal of the analysis is then to compute the posterior distribution $p(\theta|x_{1:T}, y_{1:T})$. This distribution will enable us to classify new data as follows

$$p(y_{T+1}|x_{1:T+1}) = \int_\Theta p(y_{T+1}|x_{T+1}, \theta) p(\theta|x_{1:T}, y_{1:T}) d\theta$$

   Bayes' rule gives us the following expression for the posterior

$$\begin{aligned} p(\theta|x_{1:T}, y_{1:T}) \quad \propto \quad & \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(\theta - \mu)'(\theta - \mu)\right) \\ & \times \prod_{t=1}^{T} \left[\frac{1}{1 + \exp(-\theta' x)}\right]^{y_t} \left[1 - \frac{1}{1 + \exp(-\theta' x)}\right]^{1-y_t} \end{aligned}$$

   The problem is that in this case we can't solve the normalising integral analytically. So we have to use numerical methods — in this case importance sampling — to approximate $p(\theta|x_{1:T}, y_{1:T})$. Note that we cannot sample from $p(\theta|x_{1:T}, y_{1:T})$ directly because we don't know the normalising constant. So instead we sample from a proposal distribution $q(\theta)$ (say a Gaussian) and weight the samples using importance sampling. After obtaining N samples of θ from the posterior, we can classify new data as follows

$$p(y_{T+1}|x_{1:T+1}) = \frac{1}{N} \sum_{i=1}^{N} p(y_{T+1}|x_{T+1}, \theta^{(i)})$$

   Load the data in the file `islogit.m`. Assume the following prior and proposal

$$p(\theta) = \mathcal{N}(1, 1.5)$$

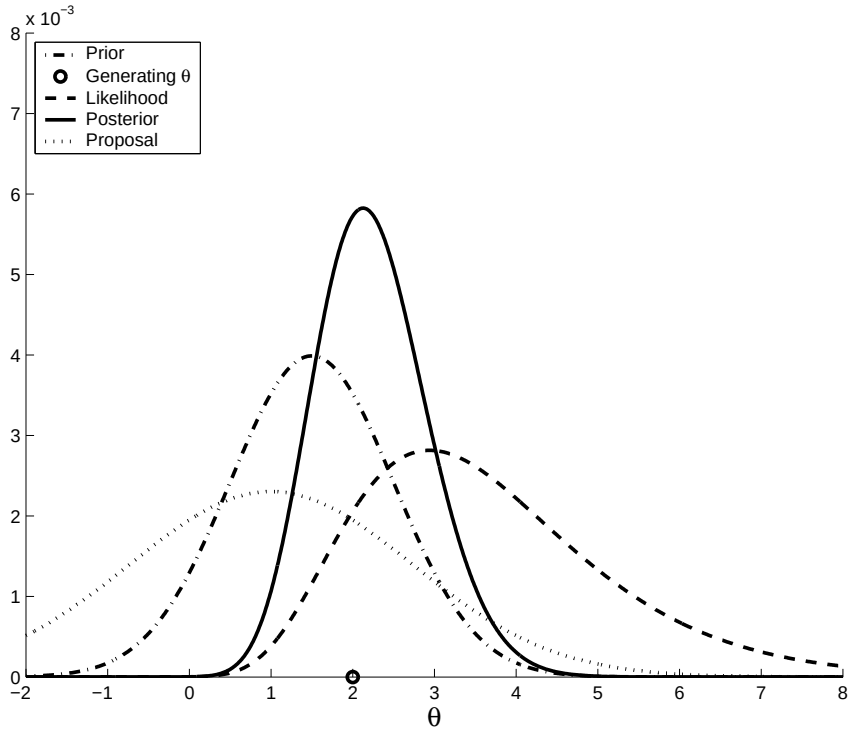


Figure 1: Plots of the distributions used in problem 1

.

$$q(\theta) = \mathcal{N}(1, 3)$$

The prior, likelihood, posterior and proposal are shown in Figure 1. I used numerical algorithms to compute them.

For this question, you need to implement an importance sampler to approximate $p(\theta|x, y)$. So you need to sample from it and then you need to use the `hist` command to produce the plot shown in Figure 2. The top histogram represents the approximation of the posterior, while the bottom one is the predictive distributions (probability of outcome being zero or one). Repeat the experiment for $N = 1000$ and $N = 10000$ samples.

To help you a bit, here's the structure of the program. Fill in the missing code where the sign ??? appears.

```
clear;

% SIMULATION PARAMETERS:
% ======================
N = 10000;              % Number of samples..
N_bins = 20;            % Number of bins in the histogram.
theta = zeros(N,1);     % Samples.
```

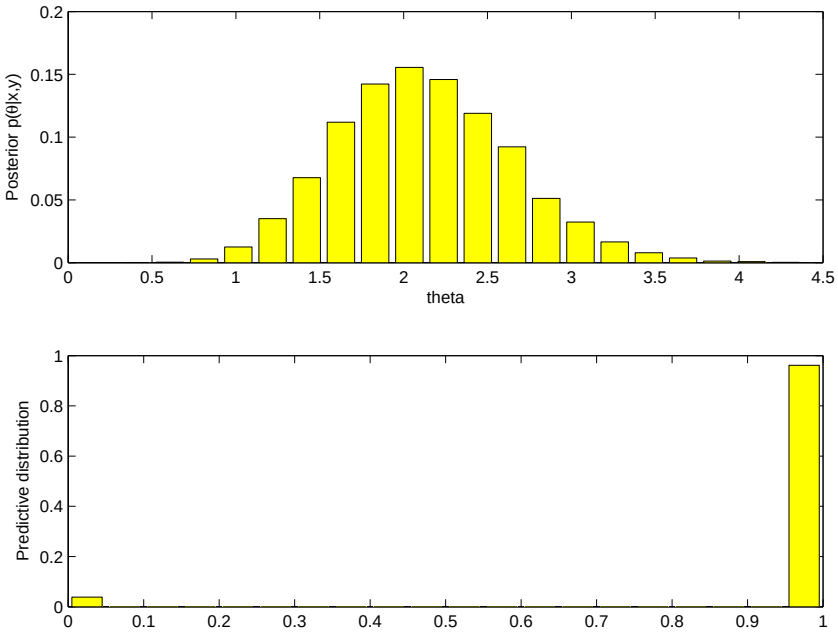


Figure 2: Solution to problem 1

.

```
% PROPOSAL SPECIFICATION:
% ======================
s_q = 3;              % Proposal variance.
m_q = 1;              % Proposal mean.

% PRIOR SPECIFICATION:
% ====================
s_p = 1;                % Prior variance.
m_p = 1.5;              % Prior mean.

% LOAD THE DATA:
% =============
islogit;
[T,arb] = size(x);

% IMPORTANCE SAMPLING:
% ====================
for i=1:N,
  z = ???
  logLikelihood = ???
  target =  ???
  proposal = ???
  w(i) = target/(proposal+1e-99);
  theta(i,:) = ???;
end;
```

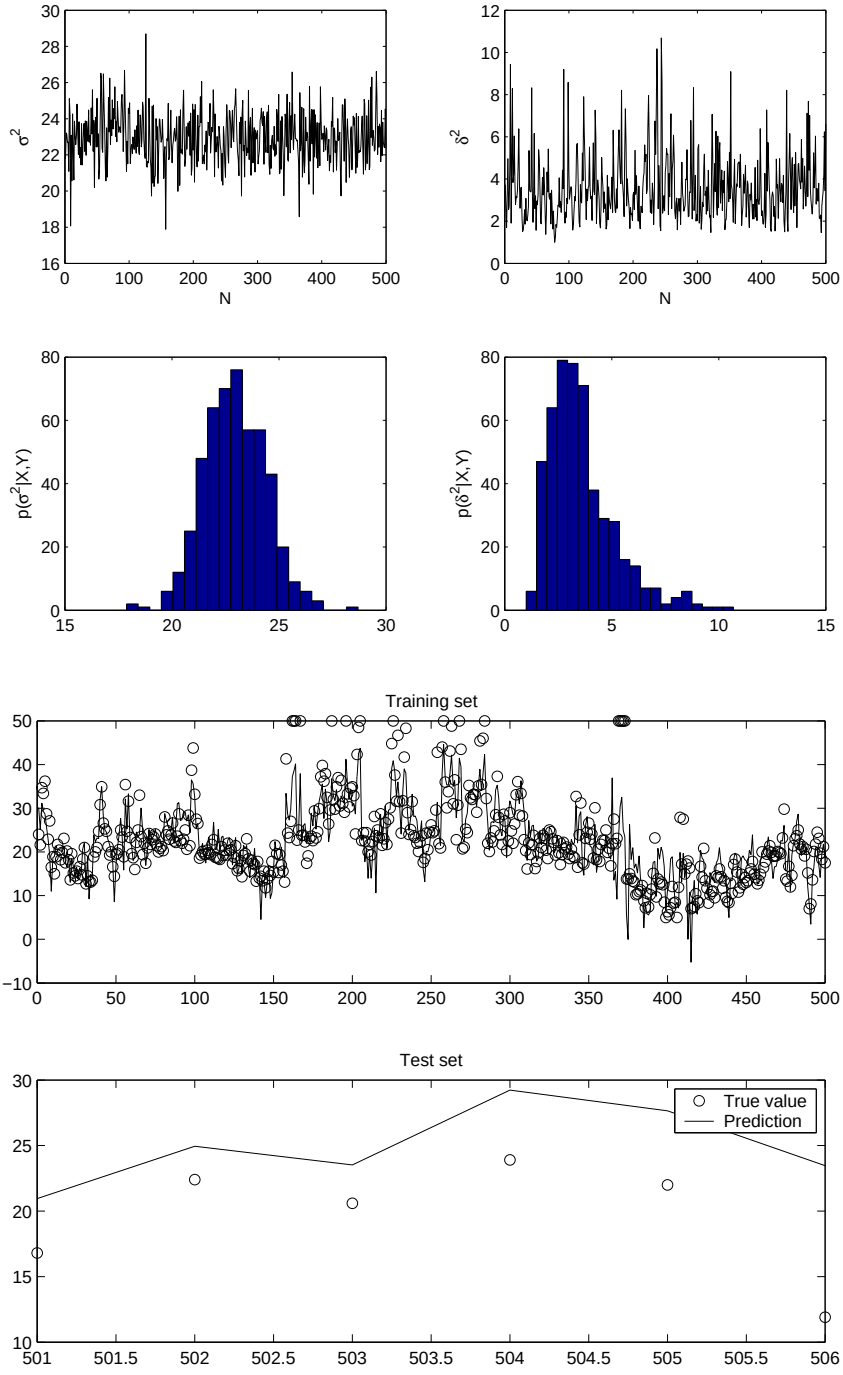
```
w = w./sum(w);           % Normalise the weights.

% RESAMPLING:
% ===========
% This is a black box routine.
resampled_index = deterministicR(1:N,w');
theta(:,:) = theta(resampled_index,:);


% PLOTS THE TRUE POSTERIOR AND THE MONTE CARLO APPROXIMATION:
% ===========================================================
figure(1)
clf;
subplot(211)
[b,a] = hist(theta,N_bins);
bar(a,b/sum(b),'y')
ylabel('Posterior p(\theta|x,y)')
xlabel('theta')

% TEST DATA:
% =========
logit_pred = zeros(N,1);
y_pred = zeros(N,1);
x = randn; % Generate a random test point.
for i=1:N,
  logit_pred(i) = (1+exp(-x*theta(i))).^(-1);
  y_pred(i) = rand < logit_pred(i); % i.e. y_pred is Bernoulli(logit_pred).
end;
subplot(212)
[b,a] = hist(y_pred,N_bins);
bar(a,b/sum(b),'y')
ylabel('Predictive distribution')
```

2. **Gibbs Sampling for Linear Regression**:

   Implement the Gibbs sampler for linear regression from the lecture notes. Apply it to the problem of predicting house prices in Boston. The code should start with the following lines:

```
echo off; clear;

% LOAD THE DATA AND EDIT IT:
% ========================

data = load('housing.data');      % Load the data.
x = data(:,[1:3 5:13]);            % Input data.
[nn,dold] = size(x);
x = [ones(nn,1) x];               % Add 1 for bias term.
[nn,d] = size(x);                 % d is the dimension of theta
                                  % nn is the total number of data
y = data(:,14);                   % Output data.

% PRIOR PARAMETERS
% ================
a = 0;
b = 0;
alpha = 2;
beta = 10;


% CREATE THE TRAIN AND TEST SETS:
% ==============================
n = 500;                % Number of training examples.
xTrain = x(1:n,:);      % Training input data.
yTrain = y(1:n,:);      % Training ouput data.
xTest = x(n+1:nn,:);    % Test input data.
yTest = y(n+1:nn,:);    % Test output data.
[n2,tmp] = size(xTest);

% INITIALIZATION:
% ===============
XY = xTrain'*yTrain;
XX = xTrain'*xTrain;
X = xTrain;
Y = yTrain;
N = 500;                % Number of samples.
theta = zeros(d,N);
mu = zeros(d,N);
sigma2 = zeros(N,1);
delta2 = zeros(N,1);
delta2(1) = inv(gengamma(alpha,beta));

for i=1:N,

???
```

Your code should be able to produce the following plots



Compare your training and test errors with the least squares solution.

3. **PCA and Gaussian Processes for Motion Capture Data**:

   In this exercise the data consists of a matrix $\mathbf{X} \in \mathbb{R}^{\mathbf{993}\times\mathbf{69}}$ of 993 captured human poses of 69 points. Each point corresponds to a position of a body part, e.g. neck, left arm, right foot and so on. The goal of the exercise is to project the 993 poses to a 2D space using PCA. Next, we construct a nonlinear Gaussian Process regression function from the 2D PCA space to the 69D body part space. That is, by specifying a point in the 2D space, the nonlinear regression model should predict a pose in the 69D space. This enables us to control a character with many degrees of freedom using a 2-dimensional interface.

   The website provides three files. The file `loaddata.m` simply loads the data. The file `defineLines.m` describes how each body part is connected to the other body parts using lines. Finally, the file `mocaph.m` is the most important one. You should edit this file and enter the right commands (where question marks appear) in order to do PCA and nonlinear regression. Note that the nonlinear regression machine is tested both on training and test data and compared to the true data.

   You should hand in all your code and a picture of the last frame of animation.

4. **K-means and EM**:

   The file `clusterData.dat` on the homework website contains 200 rows of (x_1, x_2) pairs.

   (i) Plot this data. The plots in the following parts should be plotted on top of this plot.

   (ii) Implement the K-means algorithm and apply it to the data, using 3 clusters and iterating until the algorithm converges. For initialisation, use the cluster centroids $\mu_1 = (0.5, 2.3)$, $\mu_2 = (0.5, 0.5)$ and $\mu_3 = (5.2, -0.1)$. Plot the evolution of the cluster centroids as the algorithm runs (that is, plot $\mu_c^{(t)}$ for each c and iteration t).

   (iii) Implement the EM algorithm for fitting a mixture of Gaussians,

$$p(x|\theta) = \sum_{c=1}^{3} p(c)\mathcal{N}(x|\mu_c, \Sigma_c),$$

   on the same dataset. For initialisation, use the same means as in (ii) and set $p(c) = (1/3, 1/3, 1/3)$ and $\Sigma_1 = \Sigma_2 = \Sigma_3 = I$. Plot the evolution of the means and (optional) superimpose the elipses for the variances around the respective means.

   (iv) Comment on the relative rates at which the two algorithms converge.

5. **Vector quantisation and image compression**:

   In this problem, we will apply the K-means algorithm to lossy image compression, by reducing the number of colors used in an image.

   The data website contains a 512x512 image of a mandrill represented in 24-bit colour. This means that, for each 262144 pixels in the image, there are three 8-bit numbers (each ranging from 0 to 255) that represent the green, red and blue intensity values for that pixel. The straightforward representation of this image therefore takes about 262144x3 = 786432 bytes (a byte being 8 bits). To compress the image, we will use K-means to reduce the image to K=16 colours. More specifically, each pixel in the image is considered a point in the three-dimensional (r,g,b)-space. To compress the image, we will cluster these points in colour-space into 16 clusters, and replace each pixel with the closest cluster centroid.

   (i) Load and plot the image `mandrill-large.tiff` using the following Matlab commands:

```
A = double(imread('mandrill-large.tiff'));
imshow(uint8(round(A)));
```

   The first line creates a three dimensional matrix, such that `A(:,:,1)`,`A(:,:,2)` and `A(:,:,3)` are 512x512 arrays that respectively contain the red, green and blue values for each pixel. Since this image is large, K-means would take too long. Instead, you should load and cluster the image `mandrill-small.tiff`. In particular, treat each pixel's (r,g,b) values as an element of $\mathbb{R}^3$ and run K-means with 16 clusters on this image. Iterate to (preferably) convergence, but in any case for less than 30 iterations. For initialisation, set each cluster centroid to the (r,g,b) values of a randomly chosen pixel in the image.

   (ii) Take the matrix A from `mandril-large.tiff`, and replace each pixel's (r,g,b) values with the value of the closest cluster centroid. Display the new image, and compare it visually to the original image. Hand in all your code and a printout of your compressed image next to the original image (printing on a black-and-white printer is fine).

6. **Clustering text documents**:

   Load the file `textData.dat`, available on the homework website. This file contains a document-word matrix consisting of 8 documents. Cluster the documents using a mixture of Multinomial distributions with $K = 10$ clusters. Do this by finding both the maximum likelihood and maximum a posteriori (MAP) estimates of the mixture parameters. For the MAP estimator, choose the Dirichlet prior parameters to be $\alpha = 1$ and $\beta = 4$.

   (i) For each estimator plot a histogram of the cluster probabilities $p(c)$.

   (ii) How many clusters of documents are there (roughly)? You might have to run EM a few times with different initialisations in this data exploration stage.

   (iii) Which documents belong together?

7. **Kalman Filtering**:

We consider the following dynamic state space model:

$$\begin{aligned} x_t &= Ax_{t-1} + Bw_t + Fu_t \\ y_t &= Cx_t + Dv_t + Gu_t, \end{aligned}$$

where $y_t \in \mathbb{R}^{n_y}$ denotes the observations, $x_t \in \mathbb{R}^{n_x}$ denotes the unknown Gaussian states, $u_t \in \mathcal{U}$ is a known control signal, the parameters (A, B, C, D, F, G) are known matrices and the initial mean and covariance of x_t are μ_0, Σ_0. The noise processes are *i.i.d* Gaussian: $w_t \sim \mathcal{N}(0, I)$ and $v_t \sim \mathcal{N}(0, I)$. Our model implies the continuous densities

$$\begin{aligned} p(x_t|x_{t-1}) &= \mathcal{N}(Ax_{t-1} + Fu_t, BB^{\mathrm{T}}) \\ p(y_t|x_t) &= \mathcal{N}(Cx_t + Gu_t, DD^{\mathrm{T}}). \end{aligned}$$

Load the data in the file `kalmanData.mat`, available on the course website. This file contains the parameters, observations and control signal. Using this information, implement a Kalman filter to estimate the hidden states $x_{1:T}$. Hand in a plot of the estimated state sequence and a plot showing the data $y_{2:T}$ and the one-step-ahead predictions $\widehat{y}_{2:T}$.

8. **Particle Filtering**:

This question builds on the importance sampling question of the previous homework. However, now we have a dynamical model. That is, the parameters are changing over time. It is therefore necessary to update the posterior distribution as data become available. Examples include tracking an aircraft using radar measurements, estimating a digital communications signal using noisy measurements or estimating the volatility of financial instruments using stock market data.

The dynamic model consists of three equations: the initial probability, the transition model and the observation model. The unobserved signal (hidden states or unknown parameters) $\{x_t; t \in \mathbb{N}\}$, $x_t \in \mathcal{X}$, is modelled as a *Markov process* of initial distribution $p(x_0)$ and transition equation $p(x_t|x_{t-1})$. The observations $\{y_t; t \in \mathbb{N}\}$, $y_t \in \mathcal{Y}$, are assumed to be conditionally independent given the process $\{x_t; t \in \mathbb{N}\}$ and of marginal distribution $p(y_t|x_t)$. To sum up, the model is described by

$$\begin{aligned} p(x_0) & \\ p(x_t|x_{t-1}) & \quad \text{for } t \geq 1 \\ p(y_t|x_t) & \quad \text{for } t \geq 1 \end{aligned}$$

(i) Draw the probabilistic graphical model for this problem.

So we know $p(x_0)$, $p(x_t|x_{t-1})$ and $p(y_t|x_t)$. Our aim is to estimate recursively in time the *posterior distributionp*$(x_{0:t}|y_{1:t})$ and its associated features including the marginal distribution $p(x_t|y_{1:t})$, known as the *filtering distribution.*

At any time t, the posterior distribution is given by *Bayes' theorem*

$$p\left(x_{0:t}|\, y_{1:t}\right) = \frac{p\left(y_{1:t}|\, x_{0:t}\right) p\left(x_{0:t}\right)}{\int p\left(y_{1:t}|\, x_{0:t}\right) p\left(x_{0:t}\right) dx_{0:t}}$$

(ii) Show that the following recursive update is true:

$$p\left(x_{0:t+1}|\, y_{1:t+1}\right) = p\left(x_{0:t}|\, y_{1:t}\right) \frac{p\left(y_{t+1}|\, x_{t+1}\right) p\left(x_{t+1}|\, x_t\right)}{p\left(y_{t+1}|\, y_{1:t}\right)}$$

State your conditional independency assumptions clearly.

The marginal distribution $p\left(x_t|\, y_{1:t}\right)$ also satisfies the following recursion

$$\textit{Prediction: } p\left(x_t|\, y_{1:t-1}\right) = \int p\left(x_t|\, x_{t-1}\right) p\left(x_{t-1}|\, y_{1:t-1}\right) dx_{t-1}$$

$$\textit{Updating: } p\left(x_t|\, y_{1:t}\right) = \frac{p\left(y_t|\, x_t\right) p\left(x_t|\, y_{1:t-1}\right)}{\int p\left(y_t|\, x_t\right) p\left(x_t|\, y_{1:t-1}\right) dx_t}$$

These expressions and recursions are deceptively simple as one cannot typically compute the integrals. However, if the distributions are Gaussian or discrete, we can solve the integrals. In the Gaussian case, the answer is known as the Kalman filter. In the discrete case, the answer gives rise to the HMM filter. In general, we need to do sampling to solve this problem.

Let's do importance sampling. That is, we sample from a proposal distribution that we choose and then weight the samples appropriately. Let the proposal distribution be

$$q\left(x_{0:t}|\, y_{1:t}\right) = p\left(x_{0:t-1}|\, y_{1:t-1}\right)\ q\left(x_t|\, x_{0:t-1}, y_{1:t}\right)$$

That is, the proposal only affects the state at time t. In other words, we don't modify samples in the past.

(iii) Show that the importance weights are given by

$$w_t \propto \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q(x_t|x_{1:t-1}, y_{1:t})}$$

Hint:

$$w_t = \frac{p(x_{0:t}|y_{1:t})}{q(x_{0:t}|y_{1:t})}$$

(iv) What's the expression for w_t when we go for the choice $q(x_t|x_{1:t-1}, y_{1:t}) = p(x_t|x_{t-1})$?

The idea is to draw N samples (known as particles) at each time step t. We then weight these samples and choose the fittest samples by resampling. The basic algorithm is as follows:

Simple Particle Filter

(a) *Initialization*, $t = 0$.

- For $i = 1, ..., N$, sample $\mathbf{x_0^{(i)}} \sim \mathbf{p}(\mathbf{x_0})$ and set $t = 1$.

2. *Importance sampling step*

- For $i = 1, ..., N$, sample $\widetilde{\mathbf{x}}_t^{(i)} \sim p\left(\mathbf{x_t} | \mathbf{x_{t-1}^{(i)}}\right)$.
- For $i = 1, ..., N$, evaluate the importance weights $\widetilde{w}_t^{(i)} \propto p\left(\mathbf{y_t} | \widetilde{\mathbf{x}}_t^{(i)}\right)$
- Normalise the importance weights.

3. *Selection step*

- Resample with replacement N particles $\left(\mathbf{x_t^{(i)}}; \mathbf{i = 1}, \ldots, \mathbf{N}\right)$ from the set $\left(\widetilde{\mathbf{x}}_t^{(i)}; i = 1, \ldots, N\right)$ according to the importance weights.
- Set $t \leftarrow t + 1$ and go to step 2.

The set of N particles at each time step, provides a description of the filtering distribution $p(x_y|y_{1:t})$.

For demonstration purposes, we apply the bootstrap algorithm to the following nonlinear, non-Gaussian model

$$\begin{aligned} x_t &= \frac{1}{2}x_{t-1} + 25\frac{x_{t-1}}{1 + x_{t-1}^2} + 8\cos(1.2t) + v_t \\ y_t &= \frac{x_t^2}{20} + w_t \end{aligned}$$

where $x_1 \sim \mathcal{N}\left(0, \sigma_1^2\right)$, v_t and w_t are mutually independent white Gaussian noises, $v_t \sim \mathcal{N}\left(0, \sigma_v^2\right)$ and $w_t \sim \mathcal{N}\left(0, \sigma_w^2\right)$ with $\sigma_1^2 = 10$, $\sigma_v^2 = 10$ and $\sigma_w^2 = 1$.

(v) The implementation is is the file `pfhomework.m`. Run this file with the data `trackingData.mat` and save the two plots. Finally, replace the model by the following model

$$\begin{aligned} x_t &= \frac{1}{2}x_{t-1} + 25\frac{x_{t-1}}{1 + x_{t-1}^2} + 5\sin(1.2t) + v_t \\ y_t &= \frac{x_t^2}{11} + w_t \end{aligned}$$

where $\sigma_1^2 = 2$, $\sigma_v^2 = 2$ and $\sigma_w^2 = 0.05$. Run the algorithm using the data file `trackingData2.mat`. Generate the two plots. In total, you should hand in four plots and explain the results.