Statistical Methods in Applied Computer Science, Fall 2018
Assignment 2 (actually 2,3 and Project)

Jens Lagergren, Olivier Bilenne, Hazal Koptagel, and Seong-Hwan Jun

Deadline January 15, 2018

This assignment should be done in pairs. Describe your results in a short report using at most 2 pages per problem. You will present the assignment by a written report, submitted before the deadline using Canvas. You must solve the assignment independently and it will automatically be checked for similarities to other students' solutions as well as documents on the web in general. From the report it should be clear what you have done and you need to support your claims with results. You are supposed to write down the answers to the specific questions detailed for each task. This report should clearly show how you have drawn your conclusions and explain your derivations. Your assumptions, if any, should be stated clearly.

Being able to communicate results and conclusions is a key attribute of any scientific practitioner. It is up to you as an author to make sure that the report clearly shows what you have done. Based on this, and only this, we will decide if you pass the task. No detective work should be required on our side. In particular, neat and tidy reports please! I very much recommend you to get used to LaTeX to write your report, it is a great tool.

Sumbission Instructions
All code to be implemented in Python. We have provided basic skeleton code with function signatures at https://github.com/junseonghwan/kth-stat-method. There are two directories *a2, a3p* and within each of the two directories, there are subdirectories, one for each question. You may create additional files as you wish and you are free to modify the provided files. However, function signatures should not be altered. The return types of the variables to be returned from MCMC/SMC code should be in numpy array or Python list format. We have declared the return types in the code provided, you should not modify the return types in such cases. We will randomly select a subset of questions for which we will run and test for correctness of your implementation. To that end, we will set up our scripts as follows:

```
from q1 import *
from q1generator import *

some_seed = 1903572
X, s, o = generate_data(some_seed, 100, 4, 0.05)
s, X = mh_w_gibbs(o, 1000)

# code for testing correctness follows
```

We will use automatic test script guided by the principles described in [1].

We strongly recommend that you create a private Github repository to store all your source code for seamless collaboration with your partner. You can apply for Github student pack at https://education.github.com/pack, which allows students to host unlimited number of

private repositories. The recommended approach to submission is via Github. We will inform you of our Github handles, at which point we ask you to add us as collaborators to your private repository so that we can clone and download your code. Any external libraries and dependencies should be specified in the README.md file with instructions on installing them within conda environment (https://conda.io/docs/). However, as alluded to earlier, it should not require much detective work on our part to set up any libraries. Therefore, we recommend that you limit your use of external libraries as much as possible.

Grading

E Correctly completed two of Task 2.1 and 2.2

D E + Correctly completed **one** of Task 2.3, 2.4, and 2.5.

C E + Correctly completed **two** of Task 2.3, 2.4, and 2.5.

B E + Correctly completed **three** of Task 2.3, 2.4, and 2.5 as well as **one** of 3.1 and 3.2 (which will be released around December 23)

A E + Correctly completed **three** of Task 2.3, 2.4, and 2.5 as well as **two** of 3.1 and 3.2.

These grades are valid for assignments submitted before the deadline, late assignments can at most receive the grade E.

Good Luck!

2.1 Gibbs sampler for the magic word

The following generative model generates N sequences of length M, $s^1, \ldots, s^N$ where $s^n = s^n_1, \ldots, s^n_M$. All sequences are over the alphabet $[K]$. Each of these sequences has a "magic" word of length W hidden in it and the rest of the sequence is called background.

First, for each n, a start position r_n for the magic word is sampled uniformly from $[M - w + 1]$. Then the j:th positions in the magic words are sampled from $q_j(x) = \text{Cat}(x|\theta_j)$ where θ_j has a $\text{Dir}(\theta_j|\alpha)$ prior. All other positions in the sequences are sampled from the background distribution $q(x) = \text{Cat}(x|\theta)$ where θ has a $\text{Dir}(\theta|\alpha')$ prior.

We are interested in the posterior $p(r_1, \ldots, r_N|D)$ where D is a set of sequences $s^1, \ldots, s^N$ generated by the model and r_n is the start position of the magic word in the n:th sequence s^n. The following describes a Gibbs sampler that can be used for estimating the posterior over start positions after having observed $s^1, \ldots, s^N$. The sampler is collapsed and we do know the hyperparameters α and α'. The states are vectors of start positions $R = (r_1, \ldots, r_N)$.

Notice that, for the Dirchlet-categorical (Dirchlet-multinoulli) distributions for the j:th position of the magic words, the marginal likelihood is

$$p(D_j|R) = [\Gamma(\sum_k \alpha_k)/\Gamma(N \sum_k \alpha_k)] \prod_k \Gamma(N^j_k + \alpha_k)/\Gamma(\alpha_k)$$

where N^j_k is the count of symbol k in the j:th column of the magic words, induced by R. For the background, the marginal likelihood is

$$p(D_B|R) = [\Gamma(\sum_k \alpha'_k)/\Gamma(B \sum_k \alpha'_k)] \prod_k \Gamma(B_k + \alpha'_k)/\Gamma(\alpha'_k)$$

where B is the number of background positions (i.e., $N(M - w)$) and B_k is the count of symbol k in the background, induced by R. The full conditional $p(r_n|R_{-n}, D)$ can be expressed as follows

$$\begin{aligned} & p(r_n|R_{-n}, D) \\ &= p(R_{-n} \cup r_n, D)/p(R_{-n}, D) \\ &\propto p(R_{-n} \cup r_n, D) \\ &\propto p(D|R_{-n} \cup r_n) \\ &= p(D_B|R_{-n} \cup r_n) \prod_j p(D_j|R_{-n} \cup r_n) \end{aligned}$$

Question 1: *Implement a Gibbs sampler for the magic word model described above. You are also supposed to generate synthetic data and estimate the posterior using your Gibbs sampler.*

Question 2: *Provide evidence that you have succeeded, in terms of convergence and accuracy.*

2.2 MCMC for the train

Problem Description

Imagine a railway with a single train. The map of the tracks is given by a $n \times n$ lattice where each vertex corresponds to a switch with 4 neighbors. Each switch has four connections, labeled $\{0, 1, 2, 3\}$. If the train comes from the direction of $\{1, 2, 3\}$, it always leaves through direction 0. If the train comes from direction 0, it leaves through one of the remaining three directions, depending on the switch setting. The switch setting is unknown and we place a prior probability of $1/3$ for each direction. Note that the switch setting remains the same throughout the train run (i.e., there is no operator that changes its orientation).

We represent the map of the tracks by undirected graph $G = (V, E)$, where V denotes the set of all switches and E denotes the set of edges connecting the switches. Note that each vertex has degree 4 and the graph is undirected. At each vertex the edges are labeled $\{0, 1, 2, 3\}$ (an edge can have different labels at a pair of neighboring vertices).

We receive a stream of signals from the train, each signal specifying the direction in which train has passed a switch: $\{01, 02, 03, 10, 20, 30\}$. However, you do not know which switch the train has passed. Also, the sensors are noisy, and with a certain probability $p \in [0, 1]$, the train reports a random signal rather than the real direction. We denote the stream of observed signals by $\boldsymbol{o} = (o_1, ..., o_T)$, where o_1 denotes the first signal received and o_T denotes the last signal received. Additionally, we denote the positions of the train by $\boldsymbol{s} = (s_1, ..., s_T)$ with s_1, s_T denoting the start and stop positions respectively. We place uniform prior on the start position, i.e., $s_1 = v \in V$ with probability $1/|V|$. The switch settings is viewed as a random variable, X_v for each $v \in V$, taking on values in $\{1, 2, 3\}$. We denote $\boldsymbol{X} = (X_v)_{v \in V}$ to be the set of all random variables for switch settings. We will assume that the train leaves s_1 in the direction indicated by its switch, x_{s_1}.

Note that for each signal o_t, there corresponds a position s_t. However, given the state of the switch, these positions are deterministic (i.e., if the train is at s_{t-1} and if we know $x_{s_{t-1}}$, then we know the position s_t with probability 1).

Goal: Infer the starting position of the train and the state of the switches given the observations.

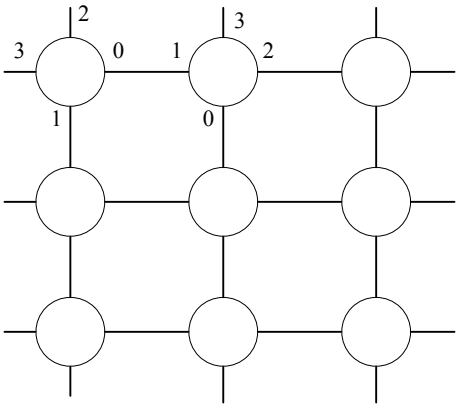


Figure 1: An example of lattice. The switches at row 1, columns 1 and 2 have their edges labelled as an illustration.

Directions

- We will sample from the posterior distribution $p(s_1, \boldsymbol{x}|G, \boldsymbol{o})$ using MCMC. We will first sample $s_1|G, \boldsymbol{o}, \boldsymbol{x}$ and then $\boldsymbol{x}|G, \boldsymbol{o}, s_1$.
- To that end, the conditional likelihood of the observations given the switch states and start position needs to be computed.

To compute the conditional likelihood, we need to marginalize out the latent positions:

$$p(\boldsymbol{o}|G, \boldsymbol{x}, s_1) = \sum_{s_2} ... \sum_{s_T} p(\boldsymbol{o}|G, \boldsymbol{x}, \boldsymbol{s}) p(\boldsymbol{s}_{2:T}|G, \boldsymbol{x}, s_1), \tag{1}$$

where $\boldsymbol{s}_{i:j} = (s_i, ..., s_j)$. Note that brute force computation quickly becomes intractable as the complexity is exponential in T. Since the train positions and the observed signals induce hidden Markov

model structure, we will exploit it to derive efficient computation of Equation 1:

$$p(\boldsymbol{o}|G,\boldsymbol{x},s_1) = \sum_{s_2} \dots \sum_{s_T} p(\boldsymbol{o}|G,\boldsymbol{x},\boldsymbol{s}) p(\boldsymbol{s}_{2:T}|G,\boldsymbol{x},s_1)$$
$$= \prod_{t=2}^{T} \sum_{s_t} p(o_t|s_t,G,\boldsymbol{x}) p(s_t|s_{t-1},G,\boldsymbol{x}). \tag{2}$$

Note that $p(s_t|s_{t-1},G,\boldsymbol{x})$ denotes the transition probability density function and $p(o_t|s_t,G,\boldsymbol{x})$ denotes the emission probability density function.

Question 3: *Implement algorithm to compute Equation 2. Clearly express the emission and transition densities and briefly describe the algorithm in words. Report runtime complexity of your algorithm and provide brief justification.*

Question 4: *Implement* Metropolis-Hastings within Gibbs *sampler for* s_1 *and* $\{X_v\}_{v\in V}$*. Specifically, implement Gibbs update for* s_1 *and Metropolis-Hastings algorithm for the swith states. Describe your sampling algorithm including i) conditional distribution for* s_1*, ii) choice of proposal distributions for* $\boldsymbol{x}$ *and the acceptance probability. Generate data and analyze it.*

Question 5: *Think about a way to assess convergence of your MCMC algorithm. Describe one such approach to assess convergence and provide evidence that you have reached convergence (using figures as appropriate).*

Question 6: *Implement Gibbs sampling that updates one variable at a time for the switch state,* $\boldsymbol{X}$*. Derive the conditional distribution of* X_v*.*

Question 7: *Implement blocked Gibbs sampling for* $\boldsymbol{X}$*. How did you determine the block size?*

Question 8: *Compare and contrast the MCMC variants implemented in this question. In particular, comment on accuracy, convergence rates, and mixing rates. Provide figures to justify your findings.*

2.3 SMC for the stochastic volatility model.

Consider the stochastic volatility model.

$$X_1 \sim \mathcal{N}(x_1|0,\sigma^2) \tag{3a}$$
$$X_t|(X_{t-1}=x_{t-1}) \sim \mathcal{N}(x_t|\phi x_{t-1},\sigma^2), \qquad t=1,\dots,T, \tag{3b}$$
$$Y_t|(X_t=x_t) \sim \mathcal{N}(y_t|0,\beta^2\exp(x_t)), \qquad t=1,\dots,T, \tag{3c}$$

where the parameter vector is given by $\theta=\{\phi,\sigma,\beta\}$ ($\phi\in\mathbb{R}$, $\sigma>0$, $\beta>0$). The latent variable X_t denotes the underlying volatility, i.e. the variations in the price of some financial asset, and Y_t denotes the observed scaled log-returns from the asset.

Question 9: *Generate parameters and data, $x_{1:T}, y_{1:T}$ from the SV model. Use $T = 100$ and fix $\phi = 1.0, \sigma = 0.16, \beta = 0.64$.*

Question 10: *Implement sequential importance sampling (SIS) to infer x_T. i) Briefly justify the choice of proposal, derive the weight update function, and a point estimate $\hat{x}_T$ of x_T. ii) Report point estimate of x_T and compute the mean squared error of the estimate to the truth as the number of samples is increased. iii) Compute the empirical variance of the normalized weights, $\boldsymbol{w}_T$. Provide density estimation plot or histogram of the weights.*

Question 11: *Implement SIS with multinomial resampling. Answer parts ii and iii from SIS question above.*

Question 12: *Implement SIS with stratified resampling. Answer parts ii and iii from SIS question above.*

Question 13: *Using 2-3 sentences, compare and contrast the three schemes.*

2.4 Stochastic volatility unknown parameters part I

Assume now that the variance parameters σ and β are both unknown, but keep $\phi = 1$.

Question 14: *Generate $x_{1:T}, y_{1:T}$ using $\beta = 0.64, \sigma = 0.16$. Make a reasonably coarse grid for β and σ between 0 and 2 (say, 8 by 8). Use the SMC with Multinomial resampling to estimate the marginal likelihood, $p_\theta(y_{1:T})$ for each pair of (β, σ) in the grid. Run SMC 10 times (each with different random seed) for each parameter combination. Choose the best parameter combination and compare it against the ground truth.*

For numerical reasons, it is better to consider the log-likelihood:

$$\log \hat{p}(y_{1:T}) = \log \prod_{t=1}^{T} \frac{1}{N} \sum_{n=1}^{N} \underbrace{p(y_t|x_t^i)}_{\tilde{w}_t^i = \alpha_t(x_{1:t}^i)} = \sum_{t=1}^{T} \left(\log \sum_{n=1}^{N} \tilde{w}_t^i - \log N \right), \tag{4}$$

where $\tilde{w}_t^i$ denotes the incremental weight of particle i at time step t.

Question 15: *Study how T and N affect the variance in the log-likelihood estimate. That is, fix T, vary N and plot the log-likelihood estimates and vice versa.*

Consider a Bayesian setting and place inverse Gamma priors on the variance parameters:

$$\sigma^2 \sim \mathcal{IG}(a = 0.01, b = 0.01), \tag{5a}$$

$$\beta^2 \sim \mathcal{IG}(a = 0.01, b = 0.01), \tag{5b}$$

where the the inverse Gamma PDF with parameters (a, b) is given by

$$\mathcal{IG}(x|a, b) = \frac{b^a}{\Gamma(a)} x^{-a-1} \exp\left(-\frac{b}{x}\right) \tag{6}$$

and Γ is the Gamma function.

Question 16: *Implement particle marginal Metropolis-Hastings sampler for this model. i) Describe your proposal distributions for β, σ, and compute the Metropolis-Hastings acceptance probability ratio. ii) Provide evidence that you have succeeded, using no more than 3 plots.*

2.5 Stochastic volatility unknown parameters part II

Since the inverse Gamma priors (5) are conjugate to the model (3), the posterior distributions of the variance parameters are available in closed form and given by

$$p(\sigma^2|\phi, x_{1:T}, y_{1:T}) = \mathcal{IG}\left(\sigma^2|a + \frac{T}{2}, b + \frac{1}{2}\sum_{t=1}^{T}(x_t - \phi x_{t-1})^2\right), \tag{7a}$$

$$p(\beta^2|\phi, x_{1:T}, y_{1:T}) = \mathcal{IG}\left(\beta^2|a + \frac{T}{2}, b + \frac{1}{2}\sum_{t=1}^{T}\exp(-x_t)y_t^2\right). \tag{7b}$$

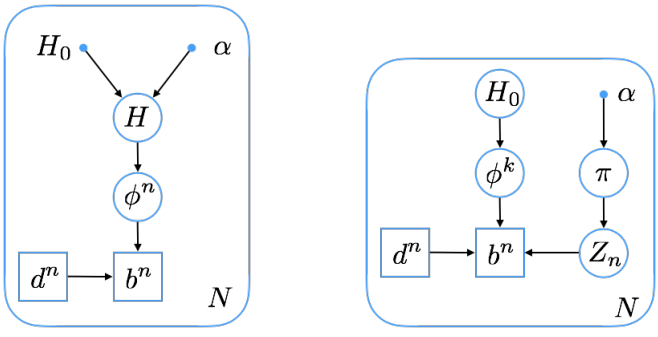
Implement a particle Gibbs sampler to compute the posterior distribution $p(\sigma^2, \beta^2|\phi, y_{1:T})$. To do so, alternately sample from

- $p(\sigma^2|\phi, x_{1:T}, y_{1:T})$ based on (7a),
- $p(\beta^2|\phi, x_{1:T}, y_{1:T})$ based on (7b),
- $p(x_{0:T}|\theta, y_{1:T}, \sigma, \beta)$ using conditional SMC.

Question 17: *Implement conditional SMC sampler for this model. Re-use the SMC implemented in 2.3 as much as possible. Report the marginal distributions of σ^2 and β^2 using histograms for the two parameters. Remember to discard the transient ('burn-in') phase of the MCMC chain. Verify that the two implemented MCMC schemes (PMMH and PG) produce the same marginal distributions for the variance parameters and compare their convergence rates.*

3.1 PyClone Light

In this problem, you should apply a collapsed Gibbs sampler in order to cluster mutations based on a Dirichlet Process Mixture Model (DPMM). The probabilistic model can, following the original publication, be described by the graphical model in Figure 2(a), where H is a Dirichlet Process (DP). We will instead use the graphical model in Figure 2(b), which yields the same distribution when equipped with distributions as follows,



(a) (b)

Figure 2: Two alternatives models for PyClone Light

$$
\begin{aligned}
\pi &\sim \text{GEM}(\alpha) \\
Z_n &\sim \pi \\
\phi^k &\sim H_0 = \text{Beta}(a_0, a_1) \\
d^n &\sim \text{Poisson}(d_0) \\
b^n &\sim \text{Bin}(d^n, \phi^{Z_n})
\end{aligned}
$$

Question 18: *Implement forward simulator to generate synthetic data that accepts hyper parameters and the number of data points as inputs.*

Question 19: *Implement collapsed Gibbs sampler for this DPMM. Test it on synthetic data with the following values of the hyper parameters:* $\alpha = 1$, $a_0 = a_1 = 1$, *and* $d_0 = 1000$. *Provide a figure depicting accuracy of clustering.*

Question 20: *The number of clusters can be controlled via the parameter* α. *Study how the performance deteriorates with increasing number of clusters. How is the number of data points affecting the performance?*

3.2 Real data analysis (mini project)

In this problem, you are asked to analyze a dataset of your interest for example, arising from your own research project. You will be asked to devise a solution using the techniques and inference methods covered in the course. Specifically, you will need to describe your problem using graphical models and one or more inference methods (e.g., particle filters, Metropolis-Hastings, or Gibbs sampler) to infer latent variables.

Question 21: *Describe your data and problem using no more than two pages (using normal font size). Provide a graphical model of your problem. Define the inference problem, in particular, clearly indicate and distinguish i) hyper parameters, ii) the observed variables, and iii) the latent variables that you would like to infer.*

Question 22: *Implement forward simulator for your graphical model. Your forward simulator should also generate hyper parameters.*

Question 23: *Devise and implement inference algorithm for the latent variables. Analyze simulated data. Report its performance using no more than two figures.*

Question 24: *Analyze real data. Report the performance of your algorithm using no more than two figures and draw any insights.*

Additional submission information for mini project

You are required to submit a file containing three functions: *hyperparam*, *generator*, and *inference*. Each function should take random seed as input. *hyperparam* is to output a numpy array containing hyper parameters generated from either prior distribution or a set of fixed values depending on your problem. Given the same seed, the output of each of the functions should be identical.

In addition to the random seed, *generator* should take in a numpy array of hyper parameters generated from *hyperparam* function and output latent variables, say $\boldsymbol{x}$, and observed variables, say $\boldsymbol{y}$. If your variables are suitably represented using numbers, you may return these as numpy arrays. Otherwise, you are welcome to declare Python Classes as necessary. Please provide sufficient in-line documentation so that the TA's can read and understand your code.

Finally, *inference* should accept the hyper parameters and the observed variables generated from *hyperparam, generator* respectively (in addition to random seed). It should output samples from the posterior distribution of the latent variables in a suitable format, say $\boldsymbol{X}$. A suitable format includes a list, list of list, and numpy array. If you have multiple latent variables, each column of $\boldsymbol{X}$ should correspond to one of the variables. Be sure to keep the index consistent across *generator* and *inference* functions, that is, i-th variable in $\boldsymbol{x}$ should correspond to i-th column in $\boldsymbol{X}$.

- If you are using particle filters, there should be a final resampling round to ensure that all particles have uniform weight or you need to return X along with the weights.
- Please keep the run time of each of your functions to not exceed 15 minutes. We will not run your code if it exceeds 30 minutes. We recommend that you specify how many minutes each of your function is expected to run.
- To that end, you may reduce the problem size. For example, if you are interested in analyzing 20 years of financial data, you may want to limit your analysis to 1 year of data. Or if you have 100 latent variables in your original problem, perhaps you can set some of them as observed or known (i.e., treat them as hyperparam) and infer only a subset of them. As long as you clearly state this in your problem description, you will not be penalized unless the problem is trivialized to an extreme extent.
- Provide in-line documentation to help the TA's in running your code as smoothly as possible.
- As the question is subjective in nature, we will evaluate your ability to clearly convey ideas, creativity, complexity of the problem, as well as correctness of implementation and easiness of execution.

Acknowledgment: Course in Sequential Monte Carlo Methods http://www.it.uu.se/research/systems_and_control/education/2017/smc.

References

[1] Samantha R Cook, Andrew Gelman, and Donald B Rubin. Validation of software for bayesian models using posterior quantiles. *Journal of Computational and Graphical Statistics*, 15(3):675–692, 2006.