IITM-CS6730 : Probabilistic Graphical Models | Release Date: Feb 28, 2020

Assignment 3 | Due Date : Mar 9, 23:59

Roll No: EE16B026 | Name: Muqeeth

Collaborators (if any): Gowri Shankar (EE16B021)

References (if any): https://people.csail.mit.edu/rameshvs/content/hmms.pdf
http://www.yindawei.com/2014/12/05/metropolis-hastings-mcmc-when-the-proposal-and-target-have-differing-support/
http://background.uchicago.edu/ whu/Courses/Ast321_11/Projects/mcmc_helsby.pdf
https://ermongroup.github.io/cs323-notes/probabilistic/gibbs/
http://nitro.biosci.arizona.edu/courses/EEB596/handouts/Gibbs.pdf

---

- Use LaTeX to write-up your solutions (in the solution blocks of the source LaTeX file of this assignment), and submit the resulting single pdf file at GradeScope by the due date. (Note: As always, **no late submissions** will be allowed, other than one-day late submission with 10% penalty! Within GradeScope, indicate the page number where your solution to each question starts, else we won't be able to grade it!).
- Collaboration is encouraged, but all write-ups must be done individually and independently, and mention your collaborator(s) if any. Same rules apply for codes written for any programming assignments (i.e., write your own code; we will run plagiarism checks on codes).
- If you have referred a book or any other online material for obtaining a solution, please cite the source. Again don't copy the source *as is* - you may use the source to understand the solution, but write-up the solution in your own words.

---

1. (10 points) [IS HMM BP IN DISGUISE? Hmmm] Consider the following Hidden Markov Model (HMM): The hidden r.v.s are $Y_0, \ldots, Y_n$ taking values in $\{1, \ldots, a\}$ and the observed r.v.s are $X_0, \ldots, X_n$ taking values in $\{1, \ldots, b\}$. The parameters are given by the transition probability matrix $T \in \mathbb{R}_+^{a \times a}$, the emission probability matrix $V \in \mathbb{R}_+^{b \times a}$ and the initial probability vector $\pi \in \mathbb{R}_+^a$. Assume all parameters are known to you. Concretely,

$$\begin{aligned} T_{j,i} &= P(Y_{t+1} = j \mid Y_t = i) \\ V_{k,i} &= P(X_t = k \mid Y_t = i) \\ \pi_i &= P(Y_0 = i). \end{aligned}$$

The joint probability is given as:

$$P(X, Y) = P(Y_0)P(X_0 \mid Y_0) \prod_{t=1}^{n} P(Y_t \mid Y_{t-1})P(X_t \mid Y_t)$$

Consider the BP algo. from class with a two-phase message-passing schedule rooted at node Y_n.

(a) (3 points) [FORWARD ALGO. FOR HMM LIKELIHOOD] Derive the evidence probability $P(X = x)$ using the BP algorithm's first phase (bottom-up or forward phase). Simplify these first-phase BP messages by expressing them as recursive updates of $\alpha_t^i := P(X_0, X_1, \ldots, X_t, Y_t = i) = \sum_{Y_0,\ldots,Y_{t-1}} P(X_0, \ldots, X_t, Y_0, \ldots, Y_{t-1}, Y_t = i)$. What are the source and target nodes of the α_t^i message, and what message is sent from the observed X_t to the hidden Y_t?
sol: given $\alpha_t^{y_t} := P(X_0, X_1, \ldots, X_t, Y_t = y_t)$

$$\begin{aligned} P(X = x) &= \sum_{y_n} P(X_0, X_1, \ldots, X_n, Y_n = y_n) \\ &= \sum_{y_n} \alpha_n^{y_n} \end{aligned}$$

$$\begin{aligned} \alpha_{t-1}^{y_{t-1}} P(Y_t \mid Y_{t-1} = y_{t-1}) &= P(X_0, \ldots, X_{t-1}, Y_{t-1} = y_{t-1}) P(Y_t \mid Y_{t-1} = y_{t-1}) \\ &= P(X_0, \ldots, X_{t-1}, Y_{t-1} = y_{t-1}) P(Y_t \mid X_0, \ldots, X_{t-1}, Y_{t-1} = y_{t-1}) \\ &= P(X_0, X_1, \ldots, X_{t-1}, Y_{t-1} = y_{t-1}, Y_t) \\ \alpha_{t-1}^{y_{t-1}} P(Y_t \mid Y_{t-1} = y_{t-1}) P(X_t \mid Y_t) &= P(X_0, X_1, \ldots, X_{t-1}, Y_{t-1} = y_{t-1}, Y_t) P(X_t \mid Y_t) \\ &= P(X_0, \ldots, X_{t-1}, Y_{t-1} = y_{t-1}, Y_t) P(X_t \mid X_0, \ldots, X_{t-1}, Y_{t-1}, Y_t) \\ &= P(X_0, X_1, \ldots, X_{t-1}, X_t, Y_{t-1} = y_{t-1}, Y_t) \\ \sum_{y_{t-1}} \alpha_{t-1}^{y_{t-1}} P(Y_t \mid Y_{t-1} = y_{t-1}) P(X_t \mid Y_t) &= \sum_{y_{t-1}} P(X_0, X_1, \ldots, X_{t-1}, X_t, Y_{t-1} = y_{t-1}, Y_t) \\ &= P(X_0, X_1, \ldots, X_{t-1}, X_t, Y_t = y_t) \\ &= \alpha_t^{y_t} \end{aligned}$$

The recursive update equation is given by $\alpha_t^{y_t} = \sum_{y_{t-1}} \alpha_{t-1}^{y_{t-1}} P(Y_t \mid Y_{t-1} = y_{t-1}) P(X_t \mid Y_t)$

The base step is $\alpha_0^{y_0} = P(X_0, Y_0 = y_0) = P(Y_0) P(X_0 \mid Y_0)$

The forward message $\alpha_t^{y_t}$ is sent from y_{t-1} to y_t

The message from the observed X_t to the hidden Y_t is $P(X_t \mid Y_t)$

(b) (2 points) [VITERBI ALGO. FOR HMM DECODING] What minimal changes will you need to make to the above algorithm so that it computes $\max_y P(X, Y = y)$ instead of $\sum_y P(X, Y = y)$. In particular, how will you change the definition of α_t^i and the corresponding recursive updates? Dont' forget to mention also how to initialize/terminate the recursive updates.
sol:
$\alpha_t^{y_t} = \max_{Y_0,\ldots,Y_{t-1}} P(X_0, \ldots, X_t, Y_0, \ldots, Y_{t-1}, Y_t = y_t)$

$\max_y P(X, Y = y) = max_{y_n} \alpha_n^{y_n}$
The recursive update equation would be

$$\alpha_t^{y_t} = \max_{y_{t-1}} \alpha_{t-1}^{y_{t-1}} P(Y_t \mid Y_{t-1} = y_{t-1}) P(X_t \mid Y_t)$$

The base step is

$$\begin{aligned} \alpha_0^{y_0} &= P(X_0, Y_0 = y_0) \\ &= P(Y_0) P(X_0 \mid Y_0) \end{aligned}$$

(c) (4 points) [BACKWARD ALGO. FOR HMM LEARNING] Simplify your second (top-down or backward) phase BP messages by expressing them as recursive updates of
$\beta_t^i := P(X_{t+1}, X_{t+2}, \ldots, X_n \mid Y_t = i)$. What are the source and target nodes of the β_t^i message? How will you use these backward messages along with the forward messages to compute $P(Y_t = i \mid X)$ (a quantity useful for learning HMM parameters from data)?
sol: given $\beta_t^{y_t} := P(X_{t+1}, X_{t+2}, \ldots, X_n \mid Y_t = y_t)$

$$\begin{aligned} \beta_{t+1}^{y_{t+1}} P(X_{t+1} \mid Y_{t+1}) &= P(X_{t+2}, \ldots, X_n \mid Y_{t+1}) P(X_{t+1} \mid Y_{t+1}) \\ &= P(X_{t+2}, \ldots, X_n \mid Y_{t+1}) P(X_{t+1} \mid X_{t+2}, \ldots, X_n, Y_{t+1}) \\ &= P(X_{t+1}, X_{t+2}, \ldots, X_n \mid Y_{t+1}) \\ \beta_{t+1}^{y_{t+1}} P(X_{t+1} \mid Y_{t+1}) P(Y_{t+1} \mid Y_t) &= P(X_{t+1}, X_{t+2}, \ldots, X_n \mid Y_{t+1}) P(Y_{t+1} \mid Y_t) \\ &= P(X_{t+1}, X_{t+2}, \ldots, X_n \mid Y_t, Y_{t+1}) P(Y_{t+1} \mid Y_t) \\ &= P(X_{t+1}, X_{t+2}, \ldots, X_n, Y_{t+1} \mid Y_t) \\ \sum_{y_{t+1}} \beta_{t+1}^{y_{t+1}} P(X_{t+1} \mid Y_{t+1}) P(Y_{t+1} \mid Y_t) &= \sum_{y_{t+1}} P(X_{t+1}, X_{t+2}, \ldots, X_n, Y_{t+1} \mid Y_t) \\ &= P(X_{t+1}, X_{t+2}, \ldots, X_n \mid Y_t) \\ &= \beta_t^{y_t} \end{aligned}$$

The recursive update equation is given by $\beta_t^{y_t} = \sum_{y_{t+1}} \beta_{t+1}^{y_{t+1}} P(X_{t+1} \mid Y_{t+1}) P(Y_{t+1} \mid Y_t)$

The base step is $\beta_n^{y_n} = 1$

The backward message $\beta_t^{y_t}$ is sent from y_{t+1} to y_t

consider the following

$$\begin{aligned}
\alpha_t^{y_t}\beta_t^{y_t} &= P(X_0, X_1, \ldots, X_t, Y_t = y_t)P(X_{t+1}, \ldots, X_n \mid Y_t = y_t) \\
&= P(X_0, X_1, \ldots, X_t \mid Y_t = y_t)P(Y_t)P(X_{t+1}, \ldots, X_n \mid Y_t = y_t) \\
&= P(X_0, X_1, \ldots, X_t \mid Y_t = y_t)P(X_{t+1}, \ldots, X_n \mid X_0, X_1, \ldots, X_t, Y_t = y_t)P(Y_t) \\
&= P(X_0, X_1, \ldots, X_n \mid Y_t = y_t)P(Y_t) \\
&= P(X_0, X_1, \ldots, X_n, Y_t = y_t) \\
&= P(Y_t \mid X_0, X_1, \ldots, X_n)P(X_0, X_1, \ldots, X_n) \\
\alpha_t^{y_t}\beta_t^{y_t} &\propto P(Y_t \mid X_0, X_1, \ldots, X_n)
\end{aligned}$$

The product of message received at Y_t will be proportional to $P(Y_t \mid X_0, X_1, \ldots, X_n)$

(d) (1 point) How many JTs (junction trees) are possible for the HMM model above, and which of these JTs would you choose to get a JT algorithm whose messages are identical to the BP algorithm messages above?
sol:
Since the product of message at node Y_t is $P(X_0, X_1, \ldots, X_n, Y_t = y_t)$, The JT for this could be n+1 cluster which are connected in chain in order of $Y_0, Y_1, .., Y_n$, ith cluster will have $Y_i, X_0, X_1, \ldots, X_n$ set of nodes in it.

2. (10 points) [JT IN THEORY] Let JT be any junction tree of a (chordal or non-chordal) MN H_Φ associated with a set of factors $\phi \in \Phi$ defined over n random variables (with mega-factors grouping these original factors denoted by $\psi_i(C_i)$).

(a) (3 points) Prove that the number of nodes in the JT is at most n. (Hint: For a given PEO of a chordal graph, assign each maximal clique to the first eliminated vertex in the clique; and count to show that the number of maximal cliques in the chordal graph is at most n.)
sol:
To construct a junction tree, we first chordal complete MN H to make H'. If the MN is already chordal, H' is same as H

Number of nodes in the JT constructed from chordal graph H' will be the number of maximal cliques in H'

We know that the number of maximal cliques in chordal graph H' is atmost n. Therefore, the number of nodes in JT will be atmost n.

To prove number of maximal cliques in chordal graph is atmost n:

Take PEO of chordal graph. As we eliminate a vertex, this vertex along with its all neighbours will be in a clique. Each clique collected in this fashion will be a maximal clique, or sub-clique of previously collected cliques.

If there is a maximal clique. It will be collected when eliminating the vertex in this cliques which comes first in PEO.

Each vertex when being eliminated cannot have two separate cliques, since the order we considered is PEO.

(b) (3 points) For any edge (C_i, C_j) in JT, let sepset $S_{ij} := C_i \cap C_j$, and let $X_{<(i,j)}$ be the set of all variables in the scope of clusters in the C_i side of the tree, and $X_{<(j,i)}$ be the set of all variables in the scope of clusters in the C_j side of the tree. Use separation criteria in H_Φ to prove that $(X_{<(i,j)} \perp X_{<(j,i)} \mid S_{ij})$ in distribution $P_\Phi(\mathcal{X})$.
sol:
$C_{<(i,j)}$ be the set of all clusters closer to i than j in JT (and hence includes C_i as well) and $C_{<(j,i)}$ be the set of all clusters closer to j than i in JT(includes C_j)

The corresponding random variables be $A_{<(i,j)} := \cup_{c \in C_{<(i,j)}} X_c$
and $B_{<(j,i)} := \cup_{c \in C_{<(j,i)}} X_c$

$X_{<(i,j)}$ is defined as $A_{<(i,j)} \setminus S_{ij}$. Similarly $X_{<(j,i)}$ be defined as $B_{<(j,i)} \setminus S_{ij}$

$X_{<(i,j)}$,S_{ij} $X_{<(j,i)}$ are disjoint. If $X_{<(i,j)}$ and $X_{<(j,i)}$ were to have a r.v common, it would be put in S_{ij} for RIP to hold. Since $X_{<(i,j)}$ and $X_{<(j,i)}$ are constructed removing S_{ij},they are disjoint.

$$\begin{aligned} P(\mathcal{X}) &= \frac{1}{Z} \prod_{c \in C} \psi_c(X_c) \\ &= \frac{1}{Z} \prod_{c \in C_{<(i,j)}} \psi_c(X_c) \prod_{c \in C_{<(j,i)}} \psi_c(X_c) \\ &= f(X_{<(i,j)}, S_{ij}) g(X_{<(j,i)}, S_{ij}) \end{aligned}$$

We know $X \perp Y \mid Z$ iff P(X,Y,Z) = f(X,Z)g(Y,Z) for disjoint X,Y,Z.Therefore $(X_{<(i,j)} \perp X_{<(j,i)} \mid S_{ij})$ is true.

(c) (2 points) At convergence (i.e., at the end of two phases of the message-passing schedule; aka (global) calibration) of the JT algorithm, prove that:

$$\mu_{ij}(S_{ij}) = m_{j \to i}(S_{ij}) m_{i \to j}(S_{ij})$$

where $\mu_{ij}(S_{ij}) := \sum_{C_i - S_{ij}} \beta_i(C_i)$ and βs are the unnormalized marginals as defined in class. Also, briefly note why $\mu_{ij}(S_{ij})$ is also equal to $\sum_{C_j - S_{ij}} \beta_j(C_j)$.
sol:
we know $\mu_{ij}(S_{ij})$ is the unnormalized potential over s_{ij}. This can be obtained by marginalizing other terms such as $c_i - s_{ij}$ in $\beta_i(C_i)$ or other terms such as $c_j - s_{ij}$ in $\beta_j(C_j)$. Where

$\beta_i(C_i) \text{and} \beta_j(C_j)$ are unnormalized potentials over c_i and c_j variables.

$$
\begin{aligned}
m_{i\to j}(s_{ij}) &= \sum_{c_i - s_{ij}} \psi_{c_i}(x_{c_i}) \prod_{k\in N(i)-j} m_{k\to i}(s_{ki}) \\
m_{j\to i}(s_{ij}) &= \sum_{c_j - s_{ij}} \psi_{c_j}(x_{c_j}) \prod_{k\in N(j)-i} m_{k\to j}(s_{kj}) \\
\beta_i(C_i) &= \psi_{c_i}(X_{c_i}) \prod_{k\in N(i)} m_{k\to i}(s_{ki}) \\
&= \psi_{c_i}(X_{c_i}) \prod_{k\in N(i)-j} m_{k\to i}(s_{ki}) m_{j\to i}(s_{ij})
\end{aligned}
$$

$$
\begin{aligned}
\mu_{ij}(S_{ij}) &= \sum_{C_i - S_{ij}} \beta_i(C_i) \\
&= \sum_{C_i - S_{ij}} (\psi_{c_i}(X_{c_i}) \prod_{k\in N(i)-j} m_{k\to i}(s_{ki}) m_{j\to i}(s_{ij})) \\
&= \sum_{C_i - S_{ij}} (\psi_{c_i}(X_{c_i}) \prod_{k\in N(i)-j} m_{k\to i}(s_{ki}))\ m_{j\to i}(s_{ij}) \\
&= m_{i\to j}(s_{ij}) m_{j\to i}(s_{ij})
\end{aligned}
$$

Hence proved.

(d) (2 points) Use above result to prove that a converged $JT = (V_{JT}, E_{JT})$ can be used to reformulate the joint distribution as:

$$\tilde{P}_\Phi(\mathcal{X}) = \frac{\prod_{i\in V_{JT}} \beta_i(C_i)}{\prod_{(ij)\in E_{JT}} \mu_{ij}(S_{ij})}.$$

sol:

$$
\begin{aligned}
\beta_i(C_i) &= \psi_{c_i}(X_{c_i}) \prod_{k\in N(i)} m_{k\to i}(s_{ki}) \\
\mu_{ij}(S_{ij}) &= m_{j\to i}(S_{ij}) m_{i\to j}(S_{ij})
\end{aligned}
$$

Consider the following

$$
\begin{aligned}
\beta_i(C_i)\beta_j(C_j) &= \psi_{c_i}(X_{c_i}) \prod_{k\in N(i)} m_{k\to i}(s_{ki})\ \ \psi_{c_j}(X_{c_j}) \prod_{k\in N(j)} m_{k\to j}(s_{kj}) \\
&= \psi_{c_i}(X_{c_i})\psi_{c_j}(X_{c_j}) m_{i\to j}(S_{ij}) m_{j\to i}(S_{ij}) \prod_{k\in N(i)-j} m_{k\to i}(s_{ki}) \prod_{k\in N(j)-i} m_{k\to j}(s_{kj})
\end{aligned}
$$

If we keep unrolling as each edge message will be multiplied in forward and backward fashion.

$$\begin{aligned}\prod_{i\in V_{JT}} \beta_i(C_i) &= \prod_{i\in V_{JT}} \psi_{c_i}(X_{c_i}) \prod_{(ij)\in E_{JT}} m_{i\to j}(S_{ij}) m_{j\to i}(S_{ij}) \\ &= \prod_{i\in V_{JT}} \psi_{c_i}(X_{c_i}) \prod_{(ij)\in E_{JT}} \mu_{ij}(S_{ij}) \\ &= \tilde{P}_\Phi(\mathcal{X}) \prod_{(ij)\in E_{JT}} \mu_{ij}(S_{ij}) \\ \tilde{P}_\Phi(\mathcal{X}) &= \frac{\prod_{i\in V_{JT}} \beta_i(C_i)}{\prod_{(ij)\in E_{JT}} \mu_{ij}(S_{ij})}\end{aligned}$$

Therefore proved.

3. (10 points) [TOYING AROUND WITH MCMC::MH] Use normal distribution centered at the current state of the chain as a proposal distribution in the Metropolis-Hastings algorithm to sample from the $\mathrm{Gamma}(\theta, 1)$ distribution when θ is a non-integer that is at least 2.

   (a) (3 points) Write down the acceptance probability (after all simplifications). What properties do you need to verify to confirm your method reaches the right stationary distribution after running for a sufficiently long time? Show your verification.
   sol:
   Let x' be the new sample and x be the current sample.
   We can follow two methods here, we can sample from $N(x, \sigma^2)$ and make proposal distribution $A(\frac{x'}{x}) = \min(1, \exp(\frac{-(x'-x)}{\theta}))$ for $x' >= 0$ and $A(\frac{x'}{x}) = 0$ for $x' < 0$.
   To demonstrate that the Metropolis-Hasting sampling generates a Markov chain whose equilibrium density is that candidate density p(x), it is sufficient to show that the Metropolis-Hasting transition kernel satisfy the detailed balance equation with p(x).
   x, y are from sample space. They are positive.
   consider $k = \exp(\frac{-(y-x)}{\theta})$
   case 1: k>1:

$$\begin{aligned}\pi(x)T(x,y) &= \pi(y)T(y,x) \\ \frac{1}{\theta}\exp(\frac{-x}{\theta})\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-(y-x)^2}{2\sigma^2})}{1}A(\frac{y}{x}) &= \frac{1}{\theta}\exp(\frac{-y}{\theta})\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-(x-y)^2}{2\sigma^2})}{1}A(\frac{x}{y}) \\ \exp(\frac{-x}{\theta})\min(1,k) &= \exp(\frac{-y}{\theta})\min(1,\frac{1}{k}) \\ 1 &= \exp(\frac{-(y-x)}{\theta})\frac{1}{k} \\ 1 &= 1\end{aligned}$$

   Hence verfied.

case 2: k<1:

$$\pi(x)T(x,y) = \pi(y)T(y,x)$$
$$\frac{1}{\theta}\exp(\frac{-x}{\theta})\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-(y-x)^2}{2\sigma^2})}{1}A(\frac{y}{x}) = \frac{1}{\theta}\exp(\frac{-y}{\theta})\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-(x-y)^2}{2\sigma^2})}{1}A(\frac{x}{y})$$
$$\exp(\frac{-x}{\theta})\min(1,k) = \exp(\frac{-y}{\theta})\min(1,\frac{1}{k})$$
$$k = \exp(\frac{-(y-x)}{\theta})$$
$$k = k$$

Hence verfied. For k = 1, equality is satisfied since x = y.

The other method is take sample only the positive ones so the proposal distribution gets normalized. Φ is cdf of standard normal distribution.
The acceptance probability of x' given x is

$$A(\frac{x'}{x}) = \min(1, \exp(\frac{-(x'-x)}{\theta})\frac{\Phi(\frac{x}{\sigma})}{\Phi(\frac{x'}{\sigma})})$$

The samples converge to required distribution if they are sampled according to the following procedure:

- sample
$$a \sim \begin{cases} \frac{N(x_t,\sigma^2)}{\Phi(\frac{x_t}{\sigma})} & \text{for a} >= 0 \\ 0 & \text{for a} < 0 \end{cases}$$
- above step is same as $a \sim N(x_t, \sigma^2)$ until a is positive.
- compute $r = A(\frac{a}{x_t})$
- set
$$x_{t+1} = \begin{cases} a & \text{with prob } r \\ x_t & \text{with prob } 1-r \end{cases}$$

To demonstrate that the Metropolis-Hasting sampling generates a Markov chain whose equilibrium density is that candidate density p(x), it is sufficient to show that the Metropolis-Hasting transition kernel satisfy the detailed balance equation with p(x).
x, y are from sample space. They are positive.
consider $k = \exp(\frac{-(y-x)}{\theta})\frac{\Phi(\frac{x}{\sigma})}{\Phi(\frac{y}{\sigma})}$

case 1: k>1:

$$\pi(x)T(x,y) = \pi(y)T(y,x)$$

$$\frac{1}{\theta}\exp(\frac{-x}{\theta})\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-(y-x)^2}{2\sigma^2})}{\phi(\frac{x}{\sigma})}A(\frac{y}{x}) = \frac{1}{\theta}\exp(\frac{-y}{\theta})\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-(x-y)^2}{2\sigma^2})}{\phi(\frac{y}{\sigma})}A(\frac{x}{y})$$

$$\exp(\frac{-x}{\theta})\frac{1}{\phi(\frac{x}{\sigma})}\min(1,k) = \exp(\frac{-y}{\theta})\frac{1}{\phi(\frac{y}{\sigma})}\min(1,\frac{1}{k})$$

$$1 = \exp(\frac{-(y-x)}{\theta})\frac{\Phi(\frac{x}{\sigma})}{\Phi(\frac{y}{\sigma})}\frac{1}{k}$$

$$1 = 1$$

Hence verfied.
case 2: k<1:

$$\pi(x)T(x,y) = \pi(y)T(y,x)$$

$$\frac{1}{\theta}\exp(\frac{-x}{\theta})\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-(y-x)^2}{2\sigma^2})}{\phi(\frac{x}{\sigma})}A(\frac{y}{x}) = \frac{1}{\theta}\exp(\frac{-y}{\theta})\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-(x-y)^2}{2\sigma^2})}{\phi(\frac{y}{\sigma})}A(\frac{x}{y})$$

$$\exp(\frac{-x}{\theta})\frac{1}{\phi(\frac{x}{\sigma})}\min(1,k) = \exp(\frac{-y}{\theta})\frac{1}{\phi(\frac{y}{\sigma})}\min(1,\frac{1}{k})$$

$$k = \exp(\frac{-(y-x)}{\theta})\frac{\Phi(\frac{x}{\sigma})}{\Phi(\frac{y}{\sigma})}$$

$$k = k$$

Hence verfied. For k = 1, equality is satisfied since x=y.

(b) (4 points) Provide your code that simulates 1000 values from $\mathsf{Gamma}(5.5, 1)$, and show the trace plots, and the histogram of X_n (with the gamma density overlaid).

```
import numpy as np
import matplotlib.pyplot as plt

def accpt_prob(x_new,x,theta,sigma):
    if(x_new <0):
        return 0
    val = np.exp(-(x_new-x)/theta)
    return min(1, val)
def get_sample(sigma, mu):
    return np.random.normal(mu,sigma)
def exact_dist(x,theta):
    return np.exp(-x/theta)/theta
```

```
def MH(n_samples,sigma,theta,x,samples):
    n = 1
    while n<n_samples:
        x_new = get_sample(sigma, x)
        p = accpt_prob(x_new,x,theta,sigma)
        temp = np.random.random_sample()
        if(temp<p):
            samples.append(x_new)
            x = x_new
        else:
            samples.append(x)
        n+=1
    return samples

theta = 5.5
sigma = 20
n_samples = 10000

x = np.random.random_sample()*100
samples = []
samples.append(x)
samples = MH(n_samples,sigma,theta,x,samples)

plt.hist(samples[15:], bins = 'auto',density = True)
x = np.linspace(0,50)
plt.plot(x,exact_dist(x,theta))
plt.legend(['gamma-density','histrogram'])
plt.show()

#for trace plots
all_samples = []
x_start = []
for i in range(5):
    x_start.append((np.random.random_sample())*100)
for i in range(5):
    x = x_start[i]
    samples = []
    samples.append(x)
    samples = MH(n_samples,sigma,theta,x,samples)
    all_samples.append(samples)
```

```
for i in range(5):
    plt.plot([i for i in range(n_samples)][0:1000],all_samples[i][0:1000])
plt.title('trace-plot-for-var=20,Burn-in=20')
plt.xlabel('iterations')
plt.ylabel('state-value')
plt.show()
```

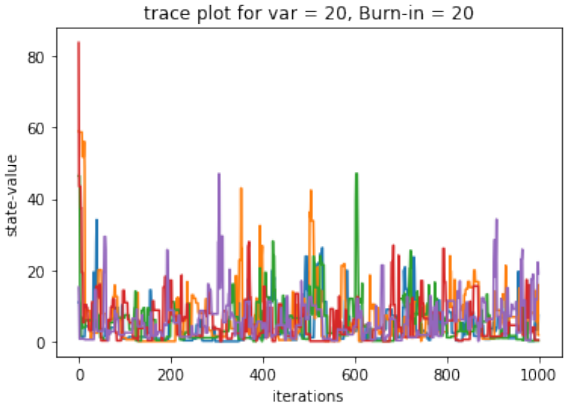


Figure 1: Trace-plot for 10000 samples

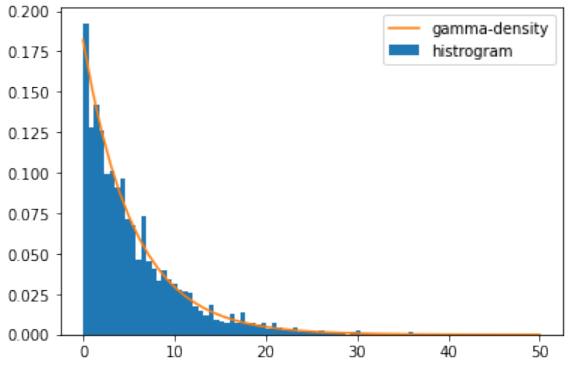


Figure 2: Histogram plot

(c) (3 points) How did you choose your burn-in time? Report it along with your acceptance rate during the burn-in vs. sample collection periods. How did these values change as a function of the variance parameter of your normal distribution, and what do you think is the optimal variance for fast convergence?
sol:
The trace plots for different start states is drawn. The time after until all chains are well mixed, is taken as burn-in time.

Acceptance rate is the ratio of number of unique values in the MCMC chain to the total number of values in the MCMC chain. The ideal acceptance is close to 23.4 percentage.

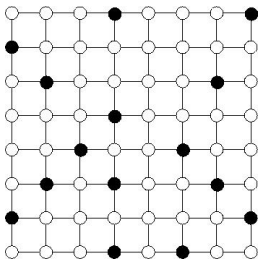
Burn-in period decreases as the variance parameter is increased and becomes constant with further increase of variance parameter.

σ^2	B	$accpt-rate-B$	$accpt-rate-sample$
10	50	0.49	0.36
15	30	0.36	0.25
18	30	0.39	0.23
20	20	0.33	0.20
30	20	0.32	0.14

The optimal variance will be in between 15 and 20 as it has acceptance rate close to ideal value

of 23.4 percentage.

4. (10 points) [HARD-CORE WITH MCMC::GIBBS]



Consider a (non-complete) connected graph *G = (V, E)* such as the one shown with $n = |V|$. Now each vertex in V gets either mapped to 0 or 1, where we only consider the following set $C \subset \{0,1\}^n$ of admissible configurations characterized by the property that pairs of adjacent vertices **cannot both** take the value 1 (see figure where black denotes 1).
Now, we want to pick one of the admissible configurations $\mathbf{x} \in C$ "at random". That is, we consider the (discrete) uniform distribution π on C, i.e. $\pi_{\mathbf{x}} = \frac{1}{|C|} \ \forall\, \mathbf{x} \in C$.

(a) (3 points) Write down the edge potentials of the undirected graphical model for this problem, and a Gibbs sampling algorithm for sampling from this model (including how you derived the associated conditional $P(X_i|X_{-i})$). Does your algorithm need to know the partition function |C|?
sol:
Consider two adjacent nodes, X_i and X_j, from the local information we can have edge potential as follows:

X_i	X_j	$\phi(X_i, X_j)$
0	0	10
0	1	10
1	0	10
1	1	0

Gibbs sampling algorithm:

- Initialize $x^t = (x_1^t, x_2^t, \ldots, x_n^t)$ for t = 0
- for t= 1,2,..,N
  - pick index i at random from $1, 2, \ldots, n$
  - draw a sample $a \sim P(X_i \mid X_{-i})$
  - $x^{t+1} = (x_1^t, \ldots, x_{i-1}^t, a, x_{i+1}^t, \ldots, x_n^t)$

From markov property we have, $P(X_i \mid X_{-i}) = P(X_i \mid Nbrs(X_i))$. We donot require to know partition function for gibbs sampling.

If $Nbrs(X_i)$ all takes 0, then

X_i	$P(X_i \mid Nbrs(X_i))$
0	0.5
1	0.5

If atleast one of $Nbrs(X_i)$ takes value 1, then

X_i	$P(X_i \mid Nbrs(X_i))$
0	1
1	0

(b) (3 points) Is the Markov chain you set up irreducible and aperiodic? Does your chain admit a distribution that satisfies detailed balance? If it simplifies your proof, assume here that your

Gibbs sampling routine employs "random scan" (pick a random i from $1, \ldots, n$ and then make a move based on $P(X_i|X_{-i})$ in each epoch) instead of "systematic scan" (cycle through all i from 1 to n in each epoch).
sol:
Markov chain is irreducible:
Consider two states a,b. Configurations of ones in a and b are different. Create a path p1 by making each individual 1 in state a to 0. Similarly create path p2 by making each individual 1 in state b to 0. The path p1 appended with reverse of path p2 results us in a path from a to b with non-zero probability. Therefore all states in our markov chain are connected. So, the markov chain is irreducible.

Markov chain is aperiodic:
For aperiodic we can show every state retains its state in next transition with non-zero probability. For each node X_i, If we look at $P(X_i \mid Nbrs(X_i))$. Each X_i can retain its value in next step with non-zero probability. This can be extended to all nodes. Therefore, complete configuration can be retained with non-zero probability.
Detailed balance:
For uniform distribution on all possible states. $\pi(x) = \frac{1}{|C|}$
Let x be state where $X_i = 0$ and y be state where $X_i = 1$. In random scan, X_i node is picked. All values of other r.v's are same as before. These will be adjacent states in state transition diagram.
case 1: $Nbrs(X_i)$ all take 0.

$$\pi(x)T(x,y) = \pi(y)T(y,x)$$
$$\frac{1}{|C|}Q(x,y)A(x,y) = \frac{1}{|C|}Q(y,x)A(y,x)$$
$$Q(x,y) = Q(y,x)$$
$$P(X_i = 1 \mid Nbrs(X_i)) = P(X_i = 0 \mid Nbrs(X_i))$$
$$0.5 = 0.5$$

case 2: atleast one of $Nbrs(X_i)$ take 1.
Here X_i cannot change its value. detailed balance equation is satisfied as x = y.
Therefore, detailed balance is satisfied by our markov chain

(c) (4 points) Provide your code and trace plots (of some functions that each map a configuration to a real value that helps visualize how well the chain is mixing). Plot the burn-in time you chose as a function of the size of the grid graphs you used. We didn't ask about the empirical acceptance rate here as it is always 100% for Gibbs sampling - prove that it is so under the same "random scan" epoch assumption above.

```
import numpy as np
```

```
import matplotlib.pyplot as plt

N = 16
grid = np.zeros(shape = (N,N))
n_samples = 10000

def check_neigh(i,j,grid):
    yes = 0
    #right
    if(i+1 <=N-1 and grid[i+1][j] ==1):
        yes = 1
    #left
    if( i-1>=0 and grid[i-1][j] == 1):
        yes = 1
    #up
    if(j-1>=0 and grid[i][j-1] == 1):
        yes = 1
    #down
    if(j+1 <= N-1 and grid[i][j+1] == 1):
        yes = 1

    return yes

#this function is useful to map each grid configuration to a real number.
def grid_to_val(grid):
    val = 0
    count = 0
    for i in range(N):
        for j in range(N):
            val+=(grid[i][j]*(pow(1.1,count)))
            count+=1
    if(val == 0):
        return 0
    val = np.log(val)
    return val

def gibbs_sampling(n_samples, grid,samples):
    for k in range(1,n_samples):
        i = np.random.randint(0,N)
        j = np.random.randint(0,N)
        if(not check_neigh(i,j,grid)):
```

```
                temp = np.random.random_sample()
                if(temp<=0.5):
                    if(grid[i][j] == 1):
                        grid[i][j] = 0
                    else:
                        grid[i][j] = 1
            samples.append(grid_to_val(grid))
        return samples

x_start = []
#all zeros in grid
x_start.append(np.zeros(shape=(N,N)))
p = np.zeros(shape=(N,N))
#diagonal ones in grid
np.fill_diagonal(np.flipud(p), 1)
x_start.append(p)
p = np.zeros(shape=(N,N))
np.fill_diagonal(p, 1)
x_start.append(p)
p = np.zeros(shape=(N,N))
for i in range(N):
    for j in range(N):
        if(i%2 == 0):
            if(j%2 == 0):
                p[i][j] = 1
            else:
                p[i][j] = 0
        else:
            if(j%2 == 0):
                p[i][j] = 0
            else:
                p[i][j] = 1
x_start.append(p)
p = np.zeros(shape=(N,N))
for i in range(N):
    for j in range(N):
        if(i%2 == 0):
            if(j%2 == 0):
                p[i][j] = 0
            else:
                p[i][j] = 1
```

```
            else:
                if(j%2 == 0):
                    p[i][j] = 1
                else:
                    p[i][j] = 0
    x_start.append(p)

    all_samples = []
    for i in range(len(x_start)):
        x = x_start[i].copy()
        samples = []
        samples.append(grid_to_val(x))
        samples = gibbs_sampling(n_samples,x,samples)
        all_samples.append(samples)

    for i in range(len(x_start)):
        plt.plot([i for i in range(n_samples)][0:5000:N],all_samples[i][0:5000
    plt.show()
```

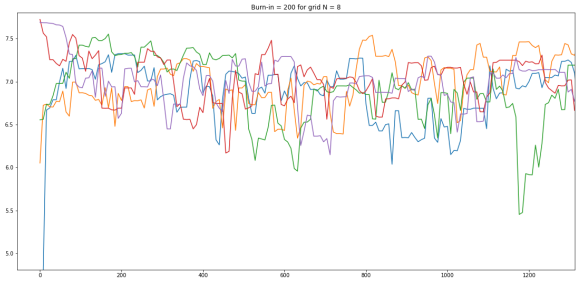


Figure 3: Trace-plot for N =8 ,B = 200

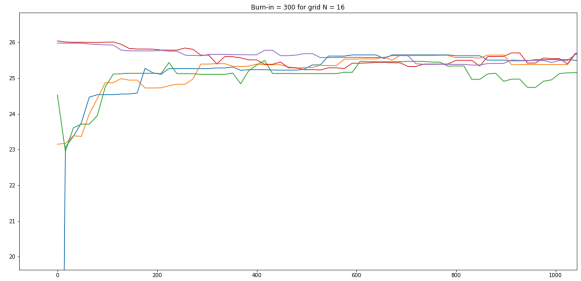


Figure 4: Trace-plot for N =16, B = 300

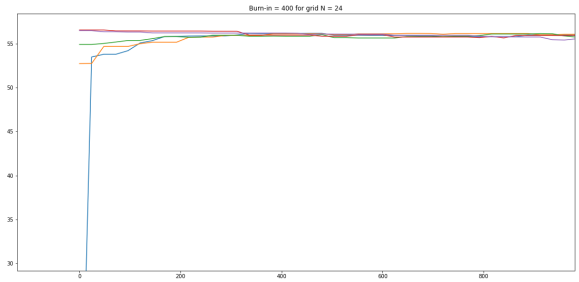


Figure 5: Trace-plot for N =32 ,B = 400

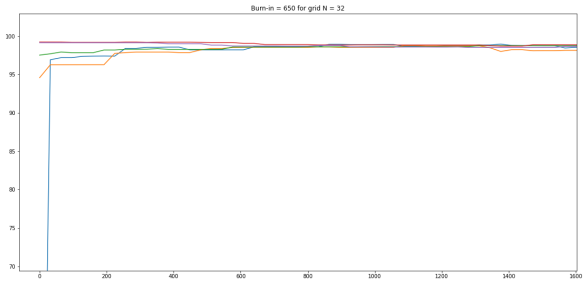


Figure 6: Trace-plot for N =16, B = 650

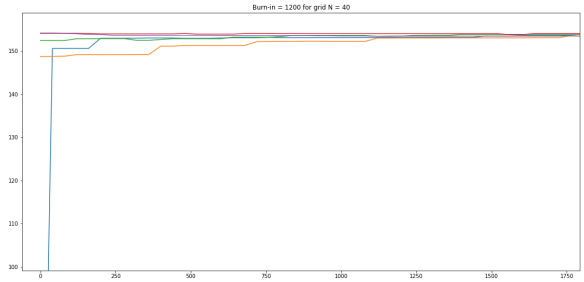


Figure 7: Trace-plot for N =40 ,B = 1300

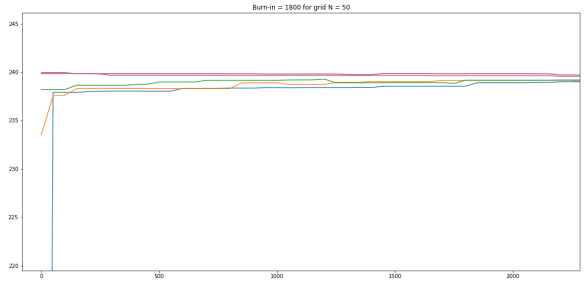


Figure 8: Trace-plot for N =50, B = 1800

Let x_i be the i^{th} random variable picked during random scan. x_{-i} be the remaining set of random variables. $Q(x'_i, x_{-i} \mid x_i, x_{-i}) = p(x'_i \mid x_{-i})$

$$\begin{aligned}\frac{p(x'_i, x_{-i})Q(x_i, x_{-i} \mid x'_i, x_{-i})}{p(x_i, x_{-i})Q(x'_i, x_{-i} \mid x_i, x_{-i})} &= \frac{p(x'_i, x_{-i})p(x_i \mid x_{-i})}{p(x_i, x_{-i})p(x'_i \mid x_{-i})} \\ &= \frac{p(x'_i \mid x_{-i})p(x_{-i})p(x_i \mid x_{-i})}{p(x_i \mid x_{-i})p(x_{-i})p(x'_i \mid x_{-i})} \\ &= 1\end{aligned}$$

Therefore, the acceptance probability is always 1.

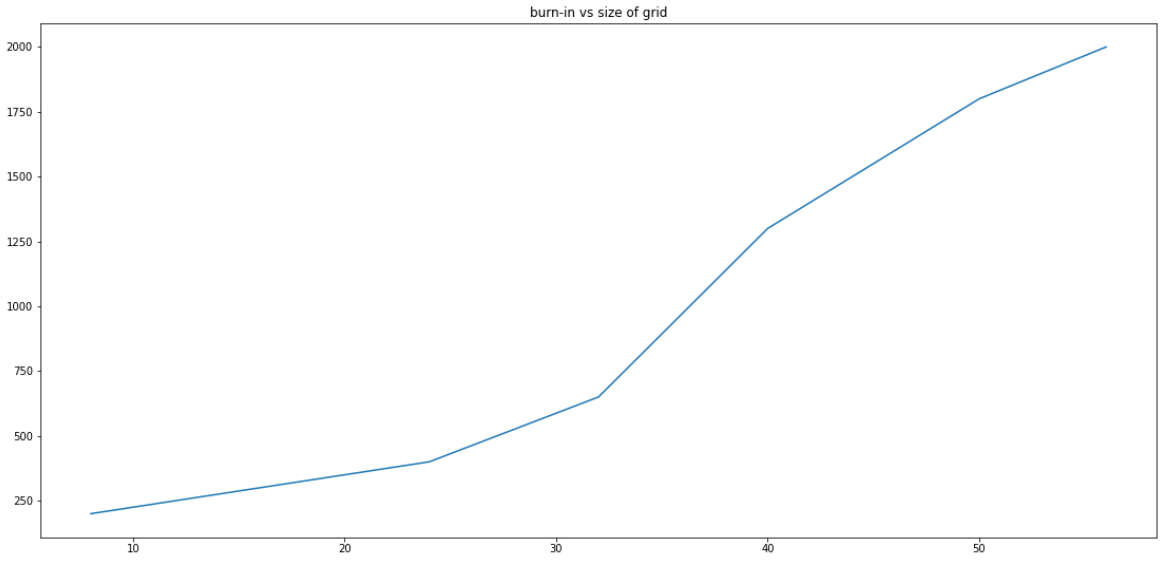


Figure 9: Burn-in period vs grid size