IITM-CS6730 : Probabilistic Graphical Models — Release Date: April 24, 2020

Assignment 4 — (Suggested) Due Date: May 3, 23:59

Roll No: PGM20XYZ — Name: Varia

Collaborators (if any):

References (if any):

- 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 suggested due date. Within GradeScope, indicate the page number where your solution to each question starts, else we won't be able to grade it!
  (**Note:** Due to the COVID situation and related institute mandates, there won't be late submission penalties to give students with poor internet access or other difficulties flexibility in completing assignments. If you do have proper internet access and other facilities, we highly recommend you to submit your solutions by the suggested due date to avoid being overwhelmed by all your works when institute reopens.)
- 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. (14 points) [CIRCULAR CLEANUP!] [from Stanford's Stat375 course]
   We want to implement a simple image segmentation/denoising algorithm based on Cluster Graph Belief Propagation (aka CG-BP or Loopy-BP) inference. The algorithm will take as input a $n \times n$ binary image encoded by the $0-1$ matrix:

$$\underline{y} = \{y_i \in \{0,1\} : i = (i_1, i_2) \in [n] \times [n]\},$$

and return a $0-1$ matrix:

$$\underline{\hat{x}} = \{\hat{x}_i \in \{0,1\} : i = (i_1, i_2) \in [n] \times [n]\}.$$

We expect nearby entries in $\hat{x}$ to have similar values; so to accomplish this goal, we introduce a graphical model of the form:

$$\mu(\underline{x}, \underline{y}) = \frac{1}{Z}\left(\prod_{i\in[n-1]\times[n]} \psi\left(x_i, x_{i+\hat{d}}\right)\right)\left(\prod_{i\in[n]\times[n-1]} \psi\left(x_i, x_{i+\hat{r}}\right)\right)\left(\prod_{i\in[n]\times[n]} \phi\left(x_i, y_i\right)\right),$$

where for $i = (i_1, i_2)$, we let $i + \hat{d} = (i_1 + 1, i_2)$ and $i + \hat{r} = (i_1, i_2 + 1)$. The compatibility functions are given in matrix form by:

$$\psi(\cdot,\cdot) = \begin{pmatrix} 1+\theta & 1-\theta \\ 1-\theta & 1+\theta \end{pmatrix}, \text{ and}$$

$$\phi(\cdot,\cdot) = \begin{pmatrix} 1+\gamma & 1-\gamma \\ 1-\gamma & 1+\gamma \end{pmatrix}$$

for some $\theta \in [0, 1]$ and $\gamma \in [0, 1]$.

(a) (3 points) Write the CG-BP (sum-product and max-product) message update rules of this model, assuming a Bethe Cluster Graph (Factor Graph) representation.

(b) (2 points) Provide a concise pseudocode that specifically shows how to initialize and schedule the message updates from above, in order to achieve the goal of constructing a denoised image $\underline{\hat{x}}$ from a noisy image $\underline{y}$.

(c) (5 points) Implement the pseudocode and test your program on a noisy image $\underline{y}$ obtained as follows: construct a 100×100 binary matrix $\underline{x}$ corresponding to a circle of radius 25 with center in position $(50, 50)$, and add noise by flipping each entry independently with probability p to obtain $\underline{y}$. Run the program on three images generated with $p = 0.1,\ 0.2$, and 0.3. Display the corresponding images (input and output), together with the values of parameters (θ, γ) used. (Note: Please also provide your denoising code that takes input/output images as comma-separated ASCII files with three columns $(i_1, i_2, value_{(i_1,i_2)})$ to help check your work).

(d) (4 points) Count the number of positions at which the estimated $\underline{\hat{x}}$ matches the noise-free circle $\underline{x}$, and average this number across many input instances/runs of your program. Report this average performance for a few values of (θ, γ), and comment on which choice offered the best performance. If you wanted to be more systematic to find the best value of (θ, γ) among all values it takes (instead of trying only a few values and taking its best), briefly mention what approach would you take?

2. (12 points) [Approximate Inference on RBMs] Consider a Markov network over m hidden $\{0, 1\}$ variables H_i and n visible $\{0, 1\}$-variables V_j. Let the joint distribution of this model be defined as:
$P(H = h, V = v) = \frac{1}{Z} \exp(h^T W v + c^T v + b^T h)$,
where $W \in \mathbb{R}_{m,n}, c \in \mathbb{R}_n n, b \in \mathbb{R}_m$ are the parameters of the model.

This is called a *Restricted* Boltzmann Machine or RBM since the edges in the Markov network are restricted to be between a hidden node and a visible node (and specifically no edges are allowed between any two hidden nodes or any two visible nodes; note that this property makes RBMs, specifically stacking of multiple RBMs, have interesting links to generative deep learning models with varied applications). To do approximate inference on a RBM model, we could employ any of these methods. Write down their update equations and simplify your terms as much as possible.

(a) (3 points) Derive standard Gibbs sampling updates of this RBM model (specifically, provide two update equations, one for sampling each H_i given all other variables and another for sampling V_j given all other variables). Report the running time complexity per update and per epoch (an epoch comprises update of every variable in the model).

(b) (3 points) Derive Block Gibbs sampling updates of this RBM model. Consider two blocks (one for all hidden node, and another for all visible nodes). Report the running time per block update - can you make this update as efficient as possible (i.e., make the running time per epoch the same as one epoch of standard Gibbs sampling) by exploiting the restricted structure of the RBM network? Can you parallelize each block update and what will be the resulting running time?

(c) (3 points) Derive the Naïve Mean Field updates of this RBM model. Describe how you can use these updates (after convergence to their fixed-point values) to approximate, specifically lower-bound, the partition function Z?

(d) (3 points) Derive the Loopy BP updates of this RBM model, again assuming a Bethe Cluster Graph (Factor Graph) representation. Express the maximized FFEF (Factored Free Energy Functional, aka Bethe free energy approximation) in terms of the converged Loopy BP messages. While this FFEF is not a theoretical lower bound for Z, argue heuristically when it can be used to approximate Z.

3. (14 points) [DIRICHLET TO OUR RESCUE] We would like to survey the rapidly accumulating research literature on the ongoing COVID-19 outbreak (for instance, a corpus dataset called CORD-19 collects 50,000+ scholarly articles about COVID-19), so as to identify and potentially address gaps in our current knowledge about the virus. To save time, you only want to have a glance at important topics and words in an article. You decided to develop a bot called Dirichlet() that can summarize you the topics present in a research article (along with the important words in varying proportions describing that topic), with no prior knowledge about what topics to look for (which suits us well as someone new to virology or epidemiology). Dirichlet() uses Latent Dirichlet Allocation (LDA) on the whole corpus of articles/documents to achieve this topic modelling task. This question is only on the theory of approximate inference in LDA (though in normal course offerings, we would also have you implement LDA using tools like gensim to analyse a corpus; interestingly, results from LDA analysis of a COVID-19 corpus is depicted here).

The LDA model can be viewed as a BN with these (local) conditional probabilities:

$$\phi_k \sim \mathtt{Dirichlet}(\beta) \tag{1}$$

$$\theta_i \sim \mathtt{Dirichlet}(\alpha) \tag{2}$$

$$z_{ji} \mid \theta_i \sim \mathtt{Categorical}(\theta_i) \tag{3}$$

$$d_{ji} \mid z_{ji}, \phi_{z_{ji}} \sim \mathtt{Categorical}(\phi_{z_{ji}}) \tag{4}$$

Here j is the index for words ($d_i = \{d_{1i}, ..., d_{Ni}\}$), i is the index for documents, and k is the index for topics. Also, we use the following notation: $N_{wki} = |\{j : d_{ji} = w, z_{ji} = k\}|$ (total number of times the word w is assigned to the topic k in document i), $N_{ki} = \sum_{w \in \text{Vocabulary}} N_{wki}$, and $N_{wk} = \sum_i N_{wki}$.

We use superscript (-ji) (e.g. N_{wki}^{-ji}) to indicate the corresponding word d_{ji} in document i is not counted in N_{wki}. By the BN chain rule, the joint distribution is thus given by:

$$\begin{aligned} P(d, z, \theta, \phi \mid \alpha, \beta) = & \quad \left(\prod_k P(\phi_k \mid \beta)\right) \prod_i P(\theta_i \mid \alpha) \prod_j P(z_{ji} \mid \theta_i) P(d_{ji} \mid z_{ji}, \phi) \\ = & \quad \left(\prod_k \frac{1}{Z(\beta)} \prod_w \phi_{kw}^{\beta_w - 1}\right) \prod_i \left(\frac{1}{Z(\alpha)} \prod_k \theta_{ik}^{\alpha_k - 1}\right) \prod_j \theta_{i, z_{ji}} \phi_{z_{ji}, d_{ji}}, \end{aligned}$$

where $Z(\alpha)$ (or $Z(\beta)$) denotes the normalizing constant of the corresponding Dirichlet distribution (based on $\Gamma(.)$ functions). Assume the hyperparameters α, β are fixed and known.

(a) (3 points) Write down $P(d \mid z, \beta)$ and $P(z \mid \alpha)$ in terms of functions of $\{N_{wk}, \beta\}$ and $\{N_{ki}, \alpha\}$ respectively. (Hint: Integrate out ϕ and θ respectively; you may find Section 4 of this writeup on Dirichlet-Categorical compound distbn. helpful.)

(b) (1 point) Exact probabilistic inference on $P(z \mid d)$ is infeasible – explain why.

(c) (5 points) Since exact inference is infeasible, we will use approximate inference. In particular, we are first interested in "collapsed" Gibbs sampling ("collapsed" since ϕ and θ are integrated out in the inference procedure). Prove the following LDA collapsed Gibbs sampling equation:

$$p(z_{ji} = k \mid z \backslash z_{ji}, d, \alpha, \beta) \propto (N_{ki}^{(-ji)} + \alpha_k) \frac{N_{wk}^{(-ji)} + \beta_w}{N_k^{(-ji)} + \sum_{w'} \beta_{w'}}$$

where w = d_{ji}.
(Hint: $\Gamma(x+1) = x\Gamma(x)$)

(d) (5 points) As an alternate approximate inference approach, we will use a variational method (in particular, Naïve Mean Field or NMF, again "(θ, ϕ)-collapsed") that approximates $P(z \mid d)$ using a fully factored distbn.:

$$Q(z \mid \lambda) = \prod_i \prod_j Q_{ji}(z_{ji} \mid \lambda_{ji}),$$

where $Q(z_{ji} = k \mid \lambda) = Q_{ji}(z_{ji} = k \mid \lambda_{ji}) := \lambda_{jik}$. The free variational parameters $\lambda = \{\lambda_{jik}\}$ that minimizes the KL-divergence $D(Q \parallel P)$ can be heuristically found using an iterative fixed-point NMF theorem seen in class. Apply this theorem to derive this NMF update rule:

$$Q_{ji}(z_{ji} = k) := \lambda_{jik} \quad \propto \quad (E_{Q_{-ji}}[N_{ki}^{(-ji)}] + \alpha_k) \frac{E_{Q_{-ji}}[N_{wk}^{(-ji)}] + \beta_w}{E_{Q_{-ji}}[N_k^{(-ji)}] + \sum_{w'} \beta_{w'}}$$

where $w = d_{ji}$ and Q_{-ji} refers to the (fully-factored) distribution over all z variables other than z_{ji} using the current iteration settings of all λ parameters other than λ_{jik}.
(Hint: You may use, $\log \frac{\Gamma(x+q)}{\Gamma(x)} = \sum_{p=0}^{q-1} \log(x+p)$, for real $x > 0$ and integer $q > 0$; and a crude approximation, $E[\log(X)] \simeq \log(E[X])$.)

(e) (0 points) *(Optional Intuition/Background Question)* A key inference query in a latent model like LDA is the conditional (posterior) probability of the hidden variables, viz., $P(z, \theta, \phi \mid d, \alpha, \beta)$, but we focused only on the "(θ, ϕ)-collapsed" probability $P(z \mid d, \alpha, \beta)$ so far, since collapsed approximate inference (collapsed Gibbs sampling or collapsed variational inference) methods are often more accurate than full (non-collapsed) approximate inference. Furthermore, once we've information on z, we can get information about (θ, ϕ) as described below.

Estimation of θ_i (document-topic proportion) and ϕ_k (topic-word distribution) are greatly simplified by knowing z_{ji} (topic assignment for each word d_{ji} in document i, drawn from $P(z \mid d)$ given by the collapsed Gibbs sampler above or from $Q(z)$ given by the collapsed variational method above). **Write down the most intuitive formula you can think for estimating $\theta_i = \{\theta_{ik}\}$ and $\phi_k = \{\phi_{kw}\}$ from all z_{ji}.** Argue why these intuitive point estimates are in fact the posterior mean of θ_i and ϕ_k (the posterior here refers to the conditional distribution $(\theta_i \mid z, d)$ or equivalently $(\theta_i \mid z)$ for θ_i, and a similar conditional distribution $(\phi_k \mid z, d)$ for ϕ_k)).

(Note: Drawing the graph structure of all three models in this assignment can aid your intuition; you can use a grid-graph for image denoising, a complete bipartite graph for RBM, and a plate network diagram for LDA.)