1. **[30 points] Deriving Gibbs Sampling for LDA.**

In this question we derive two Gibbs sampling algorithms for latent Dirichlet allocation (LDA). Recall LDA is a topic model—multiple mixture models with shared components—with the following conditional probabilities:

\[
\theta_d \mid \alpha \sim \text{Dirichlet}(\alpha, \ldots, \alpha) \quad (1) \\
\phi_k \mid \beta \sim \text{Dirichlet}(\beta, \ldots, \beta) \quad (2) \\
z_{id} \mid \theta_d \sim \text{Discrete}(\theta_d) \quad (3) \\
x_{id} \mid z_{id}, \phi_k \sim \text{Discrete}(\phi_k) \quad (4)
\]

Assume our data consists of \(D\) documents, a vocabulary of size \(W\), and we model with \(K\) topics. Let \(A_{dk} = \sum_z \delta(z_{id} = k)\) be the number of \(z_{id}\) variables taking on value \(k\) in document \(d\), and \(B_{kw} = \sum_d \sum_z \delta(x_{id} = w) \delta(z_{id} = k)\) be the number of times word \(w\) is assigned to topic \(k\). Let \(N_d\) be the total number of words in document \(d\) and let \(M_k = \sum_w B_{kw}\) be the total number of words assigned to topic \(k\).

(a) Write down the joint probability over the observed data and latent variables, expressing the joint probability in terms of the counts \(N_d, M_k, A_{dk}, \text{ and } B_{kw}\). [4 points]

(b) Derive the Gibbs sampling updates for all the latent variables and parameters. [10 points]

(c) Integrate out the parameters \(\theta_d\)'s and \(\phi_k\)'s from the joint probability in (a), resulting in a joint probability over only the \(z_{id}\) topic assignment variables and \(x_{id}\) observed variables. Again this expression should relate to \(z_{id}\)'s and \(x_{id}\)'s only through the counts \(N_d, M_k, A_{dk}, \text{ and } B_{kw}\). [6 points]

(d) Derive the Gibbs sampling updates for \(z_{id}\) with all parameters integrated out. This is called **collapsed Gibbs sampling**. You will need the following identity of the Gamma function: \(\Gamma(1 + x) = x\Gamma(x)\) for \(x > 0\). [10 points]

2. **[70 points] Implementing Gibbs sampling for LDA.** Take a look at the accompanying code, which sets up a framework in which you will implement both the standard and collapsed Gibbs sampling inference for LDA. Read the README which lays out the **MATLAB** variables used.
(a) Implement both standard and collapsed Gibbs sampline updates, and the log joint probabilities in question 1(a), 1(c) above. The files you need to edit are stdgibbs_logjoint, stdgibbs_update, colgibbs_logjoint, colgibbs_update. Debug your code by running toyexample. Show sample plots produced by toyexample, and attach and document the MATLAB code that you wrote. [10 points each]

(b) Based upon the plots of log predictive and joint probabilities produced by toyexample, how many iterations do you think are required for burn-in? Discarding the burn-in iterations, compute and plot the autocorrelations of the log predictive and joint probabilities for both Gibbs samplers. You will need to run toyexample for a larger number of iterations to reduce the noise in the autocorrelation. Based upon the autocorrelations how many samples do you think will be need to have a representative set of samples from the posterior? Describe what you did and justify your answers with one or two sentences. [10 points]

(c) Based on the computed autocorrelations, which of the two Gibbs samplers do you think converge faster, or do they converge at about the same rate? If they differ, why do you think this might be the case? Justify your answers. [10 points]

(d) Try varying $\alpha$, $\beta$ and $K$. What effects do these have on the posterior and predictive performance of the model? Justify your answers. [10 points]


(a) The functions provided for computing log predictive probabilities only compute log predictive probabilities based on the current sample, instead of averaging across samples. What are the mean log predictive probabilities of the two Gibbs samplers (discarding burn-in)? Which one is higher? Why do you think this is the case? How can we change the code of the lower one to mitigate this problem? Describe what you did and attach code? [10 points]

(b) Edit the code for computing log predictive probabilities so that they compute the log predictive probabilities averaged over samples properly. Are the computed log predictive probabilites the same now? [5 points]

4. [Bonus: 25 points] Topic modelling of NIPS papers. Now that we have code for LDA, we can try our hands on finding the topics at a major machine learning conference (NIPS)!

In the provided code there is a file nips.data which contains preprocessed data. The vocabulary is given in nips.vocab.

(a) The data in nips.data is probably too big so that our MATLAB implementation will be too slow. We will try to reduce the data set to a more tractable size, by removing words from the vocabulary. Come up with a metric for how informative/relevant/topical a vocabulary word is. You may want to experiment and try multiple metrics, and make sure that keywords like “Bayesian”, “graphical”, “Gaussian”, “support”, “vector”, “kernel”, “representation”, “regression”, “classification” etc have high metric. Report on your experiences, and use your metric to prune the data set to just the top few hundred words (say 500, or lower if the implementation is still too slow). You may find it useful to read up on tf-idf on wikipedia. [10 points]

(b) Now run LDA on the reduced NIPS data, using one of the Gibbs samplers you have just written. You will need to experiment with various settings of $\alpha$, $\beta$ and $K$ until the topics discovered looks “reasonable”. Describe the topics you found. How do the topics change (qualitatively) as $\alpha$, $\beta$ and $K$ are varied? [15 points]