Probabilistic Machine Learning (CS772A)
Homework 3 (Due date: Oct 23, 2017, 11:59pm)

Instructions

- We will only accept electronic submissions and the main writeup must be as a PDF file. If you are handwriting your solutions, please scan the hard-copy and convert it into PDF. Your name and roll number should be clearly written at the top. In case you are submitting multiple files, all files must to be zipped and **submitted as a single file** (named: your-roll-number.zip). Please do not email us your submissions. Your submissions have to be uploaded at the following link: `https://tinyurl.com/y7q2pew5`).
- Each late submission will receive a 10% penalty per day for up to 3 days. No submissions will be accepted after the 3rd late day.

Problem 1 (25 marks)

(Gibbs Sampling for Count Matrix Factorization) Consider data in form of an $N \times M$ matrix $\mathbf{X}$ where each matrix entry X_{nm} is non-negative count-valued. An example of such data would be an $N \times M$ document-word count matrix where N is the total number of documents and M is the total number of words in the vocabulary. Here X_{nm} denotes the frequency of word m in document n. Let us assume the matrix $\mathbf{X}$ to be fully observed, i.e., each entry of $\mathbf{X}$ is known. Also note that many of the entries in $\mathbf{X}$ will be zero in practice (as each document consists of a small number of words from the entire vocabulary), so $\mathbf{X}$ is usually a highly sparse matrix.

Since the data is count-valued, we would like to model it using a Poisson likelihood model instead of a Gaussian likelihood model (the model we saw in the class). A generative model for this data would be as follows

$$\begin{aligned} X_{nm} &\sim \text{Poisson}(\boldsymbol{u}_n^\top \boldsymbol{v}_m) \quad \forall n, m \\ \boldsymbol{u}_{nk} &\sim \text{Gamma}(a_u, 1/b_u) \quad \forall n, k \\ \boldsymbol{v}_{mk} &\sim \text{Gamma}(a_v, 1/b_v) \quad \forall m, k \end{aligned}$$

In the above generative model, $\boldsymbol{u}_n^\top \boldsymbol{v}_m \in \mathbb{R}_+$ denotes the rate parameter of the Poisson distribution which generates the count X_{nm} . We can also write $X_{nm} \sim \text{Poisson}(\sum_{k=1}^K u_{nk} v_{mk})$ and we can collectively write $\mathbf{X} \sim \text{Poisson}(\mathbf{U}\mathbf{V}^\top)$ where $\mathbf{U} \in \mathbb{R}_+^{N \times K}$ and $\mathbf{V} \in \mathbb{R}_+^{M \times K}$.

The above generative model assumes the shape and scale parametrization of the gamma distribution (in the definitions above, $1/b$ denotes the scale parameter of gamma). Assume the hyperparameters a_u, b_u, a_v, b_v to be fixed. Please note that gamma distribution can be defined either using the shape-rate or shape-scale parameterization (`https://en.wikipedia.org/wiki/Gamma_distribution`). Here, we will assume the shape-scale parameterization for the gamma.

For this model, derive Gibbs sampling equations for the latent factors u_{nk} and v_{mk}, and give a sketch of the overall Gibbs sampler for the model (as a pseudo-code; Problem 2 requires you to actually implement it :)).

Note: To derive the local conditionals needed by the Gibbs sampler, you may need to use the following facts:

- If $y \sim \text{Poisson}(\eta)$ where $\eta = \sum_{k=1}^K \eta_k$ then $y = \sum_{k=1}^K y_k$ where $y_k \sim \text{Poisson}(\eta_k)$, which is the Poisson additivity property. This property allows expressing a count as a sum of smaller latent counts.
- $[y_1, y_2, \ldots, y_K]|y \sim \text{multinomial}(y; \left[\frac{\eta_1}{\eta}, \frac{\eta_2}{\eta}, \ldots, \frac{\eta_K}{\eta}\right])$. This allows easily sampling the latent counts, given the total count y and $\eta_1, \ldots, \eta_k$ where $\eta = \sum_{k=1}^K \eta_k$.

Problem 2 (30 marks)

Note: This problem requires you to implement a Gibbs sampler for a model, and much of what is described below is basically instructions. :) But please do read these carefully before starting to work on this problem.

(Implementing Gibbs Sampler for Count Matrix Factorization) Your task is to implement the Gibbs sampler for the matrix factorization model described in Problem 1. The data you will use is a pre-processed version of the **20 Newsgroups** data. The data given to you is in form of a document-word count matrix of size 11284×2000 where $N = 11284$ denotes the number of documents and $M = 2000$ denotes the number of words from the vocabulary (this is a stripped down version of the original 20 Newsgroups data after removing infrequent/stop words, etc.). You are also provided the list (`vocab.txt`) of these 2000 words (you will need this list in a qualitative analysis experiment). Note that, in the provided data, each document is also associated with a label $y \in \{1, \ldots, 20\}$ but, for the purpose of this assignment, you do not have to use these labels. The dataset is given as a file named `20newsgroups.mat` (it is in MATLAB format). Loading it in MATLAB will show you the matrix $\mathbf{X}$ as well as the list of words in the vocabulary, and the the label vector $\boldsymbol{y}$ for all the documents.

You are provided some MATLAB boilerplate code (`gamma_poisson.m`) to help you get started. It includes the basic steps needed for the implementation (loading data, initialization, splitting the data into **training/test sets**, etc.) and some instructions to complete the rest of the implementation. Even if you are using some other language for your implementation (e.g., Python, R), the boilerplate should still be helpful (though I encourage using MATLAB if you are familiar with it).

Please note that you are free to use standard functions for drawing random number from standard distributions such as gamma (and other distributions). Most programming languages have in-built functions to draw samples from standard distributions (e.g., MATLAB has the `gamrnd` function which can be used to draw a gamma distributed random number for a specified shape and scale parameter).

Your Gibbs sampler would start with an initialization of the unknowns we want to sample $(\mathbf{U}, \mathbf{V})$ and, *in each iteration of Gibbs sampler*, would sample all of these from their respective local conditionals (which you would have derived for Problem 1). You might (actually will :)) need to sample a few additional variables (your solution to Problem 1 should tell you what are these variables and what will be their local conditionals).

Note: One of the key aspects of the Poisson matrix factorization model (and, in general, working with models that have Poisson likelihoods) is that the posterior doesn't depend on entries in $\mathbf{X}$ that are equal to zero (you will notice it when deriving equations in Problem 1). Therefore, the zeros in $\mathbf{X}$ can be ignored when implementing your sampler. This makes inference in models with Poisson likelihood very fast especially if the data consists of many observations that are equal to zero. Therefore, in the boilerplate code, I have extracted the nonzero entries of $\mathbf{X}$ (along with their row/column indices). Your code is supposed to use only these nonzero entries.

Training vs Test Data: I further split the nonzero entries of $\mathbf{X}$ into two sets: training and test. The train/test split ratio is 80%-20%. You are only supposed to use the training data (randomly chosen 80% of all the nonzero entries of the matrix) when doing sampling, and you would *evaluate* your model on the test data (i.e., the remaining 20% of the entries). Make sure that none of the test entries are being used in the posterior distributions of any of the variables that you are sampling. Only training entries should be used in these distributions.

Evaluation: Our evaluation criteria will be the **mean absolute error (MAE)** in matrix reconstruction. To do this, we will compute the *prediction* $\hat{X}_{nm}$ for each nonzero entry in the matrix, take its absolute difference from the actual ground truth X_{nm}, and compute the average of this quantities over all observations in the matrix $\mathbf{X}$. We will compute two such quantities: `mae_train` and `mae_test`. The former is computed over the training entries and the latter over the test entries in $\mathbf{X}$. For a correct implementation, both these numbers should typically decrease with increasing iterations (also, the test MAE would typically be slightly larger than train MAE).

How to compute $\hat{X}_{nm}$? Since $X_{nm} \sim \text{Poisson}(\boldsymbol{u}_n^\top \boldsymbol{v}_m)$ and the mean of a Poisson random variable is simply its rate parameter, we can approximate $\hat{X}_{nm}$ by the rate $\boldsymbol{u}_n^\top \boldsymbol{v}_m$. When doing sampling, we can compute this either

using a single sample of $\boldsymbol{u}_n$ and $\boldsymbol{v}_m$ (can be done using their samples from the current iteration) or via Monte Carlo averaging using all the *post burn-in* samples collected so far (also see the boilerplate code). You need to compute the MAE mentioned above using both these methods. In general, if you have implemented things correctly, the Monte Carlo averaging should give (at least slightly) lower MAE than the case when you use a single sample (especially if you have run the sampler for a significant number of iterations).

Some Qualitative Analysis and Interpretation: This matrix factorization gives us two matrices: $\mathbf{U}$ of size $N \times K$ and $\mathbf{V}$ of size $M \times K$. Note that both matrices consist of only non-negative entries (note the gamma priors) and we have essentially learned a **non-negative matrix factorization** of the count-valued matrix $\mathbf{X}$. An interesting/useful interpretation of $\mathbf{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_K]$ is in terms of each of its K columns (of size $M \times 1$) representing a "topic". For any column $\boldsymbol{v}_k, k = 1, \ldots, K$, which represents the k-th topic, you can think of entry v_{mk} telling us how prominent word m is in that topic. We can use this interpretation to find the most prominent words in each topic as follows: For each $\boldsymbol{v}_k \in \mathbb{R}_+^M$, find its 20 largest entries and print the corresponding words that are associated with these entries (the list of all the words is provided in the dataset). These words can be seen as representative of that topic. I have provided the code to do this in MATLAB (it takes the $\mathbf{V}$ matrix from the final iteration of the Gibbs sampler and produces the list of 20 most prominent words for each column of $\mathbf{V}$ (in their descending order of importance as measured by the magnitudes of v_{mk}). If you are implementing in a different language (e.g., Python or R), you will have to do the same using the method I have described.

To summarize, here are the things that you are supposed to do for this problem

- Implement the Gibbs sampler, either in MATLAB or in any other language of your choice. In either case, do refer to the provided boilerplate code for instructions and to see what steps are needed to implement the Gibbs sampler for this problem. Please use the same values for the hyperparameters (K, gamma hyperparameters, etc.) as given in the boilerplate code (if you want, you can play with these values but for the purpose of your final submission, you must use the provided values of these hyperparameters).
- In each iteration of the Gibbs sampler, you also have to compute the MAE on training as well as test entries (I have described the procedure above). You must compute the MAE using both methods mentioned above (single sample based as well as Monte Carlo based) and report the numbers in your final submission. In general, the MAE with Monte Carlo averaging would be smaller.
- Print the list of most prominent words for each column of $\mathbf{V}$ (using the procedure described above).

Submit the following: (1) Your MATLAB/Python/R code; (2) In your main writeup, include the plot of test MAE vs iterations (both using single sample as well as Monte Carlo averaging); (3) In your main writeup, include the list of 50 most prominent words for each topic $k = 1, \ldots, K$. Do any of these lists of words for each topic make sense, i.e., do they actually seem to correspond to some "topic"? Do any of these lists NOT make sense (i.e., its has mixed types of words)? Note that we are using a small K ($K = 20$ in the boilerplate code) but you should still be able to discover some topics that do make sense. Although not required for the problem, if you have time, you can play with increasing K and see if you can discover other interesting topics (the code gets slower though).

A few other things to keep in mind when implementing the Gibbs sampler for this model:

- Wherever possible, try to vectorize the code to make it run faster. In many steps of your implementation, it will usually be possible. For example, when sampling for u_{nk}, you should be able to sample for the entire vector $\boldsymbol{u}_n$ at once. E.g., even though gamma distribution is univariate, you can sample multiple gamma random variables, each with its own gamma hyperparameters, all at once (sampling functions for standard distributions in MATLAB, Python, R, etc., usually provide this functionality).
- The boilerplate code uses a total of 1000 iterations (with 500 burn-in iterations). With a correct implementation, you might find that the MAE "appears" to stabilize after 100 or so iterations, but you should let it run long enough (basically, the whole 1000 iterations) for computing the final MAE scores (and list of words in each topic). But while debugging/initial testing, you can try running for fewer iterations.

Problem 3 (20 marks)

(Implementing MH Sampling for 2-D Gaussian) In this problem, your task is to implement MH sampling to generate random samples from a 2-D Gaussian $p(\boldsymbol{z}) = \mathcal{N}\left(\begin{bmatrix}4\\4\end{bmatrix}, \begin{bmatrix}1 & 0.8\\0.8 & 1\end{bmatrix}\right)$.

To sample from $p(\boldsymbol{z})$, you will use a proposal distribution $q(\boldsymbol{z}^{(t)}|\boldsymbol{z}^{(t-1)}) = \mathcal{N}\left(\boldsymbol{z}^{(t-1)}, \begin{bmatrix}\sigma^2 & 0\\0 & \sigma^2\end{bmatrix}\right)$ and play with different values of the proposal distribution's variance σ^2. For generating the candidate sample from the proposal distribution, you can use existing functions from MATLAB/Python/R (whichever language you are using for the implementation), but you must not use the actual distribution $p(\boldsymbol{z})$ to generate the samples.

You will experiment with the following values of $\sigma^2 : 0.01, 1, 100$. For each of these cases, run the MH sampler long enough to collect 10,000 samples and show the plots of the generated samples on a 2-D plane for 100 samples, 1000 samples, and 10,000 samples (similar to the plots of slide 14, lecture-15).

Looking at the plots, which of the 3 proposals ($q(\boldsymbol{z}^{(t)}|\boldsymbol{z}^{(t-1)})$ with $\sigma^2 : 0.01, 1, 100$) seems the best choice to you? What is the rejection rate in each of these cases (rejection rate is the ratio of number of samples accepted (10,000) and the total number of candidate samples generated)? Submit the code as well as the plots.

Problem 4 (25 marks)

(Mixture of Gaussians with Missing Data) In this problem, you will re-attempt the bonus problem from the mid-sem exam (though the marks you get for this problem won't be added to your mid-sem score :)). Basically, your task is to derive the EM algorithm for a K component mixture of Gaussians when each observation $\boldsymbol{x}_n$, $n = 1, \ldots, N$ has an observed part $\boldsymbol{x}_n^{obs}$ and a missing part $\boldsymbol{x}_n^{miss}$.

- Write down the expression of the complete data log likelihood (CLL) and identify all the expectations that you will require in the expected CLL expression.
- How would you compute these expectations in the E step? In particular, give the required expressions.
- Derive the M step parameter updates for all the model parameters.

,