Assignment 6

Probabilistic and Unsupervised Learning

Maneesh Sahani

Due: 5pm Thursday December 18, 2014

Note: all assignments for this course are to be handed in to the Gatsby Unit, **not** to the CS department. Please hand your completed assignment to Barry Fong in the Alexandra House 4th floor reception, or to an instuctor to TA. Do not leave them with anyone else.

Please attempt the first questions before the bonus ones. This is a programming assignment and might require more time to understand the accompanying code and to debug, so please START EARLY.

1. **[35 points] Deriving Gibbs Sampling for LDA**.

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

$$\boldsymbol{\theta}_d|\alpha \sim \text{Dirichlet}(\alpha, \dots, \alpha) \tag{1}$$
$$\boldsymbol{\phi}_k|\beta \sim \text{Dirichlet}(\beta, \dots, \beta) \tag{2}$$
$$z_{id}|\theta_d \sim \text{Discrete}(\theta_d) \tag{3}$$
$$x_{id}|z_{id}, \boldsymbol{\phi}_{z_{id}} \sim \text{Discrete}(\phi_{z_{id}}) \tag{4}$$
$$\tag{5}$$

   Assume that our data comprises D documents with a vocabulary of size W, and that we choose to use a model with K topics. Let $A_{dk} = \sum_i \delta(z_{id} = k)$ be the number of z_{id} variables taking on value k in document d, and $B_{kw} = \sum_d \sum_i \delta(x_{id} = w)\delta(z_{id} = k)$ 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}, and B_{kw}. [5 points]

   (b) Derive the Gibbs sampling updates for all the latent variables $\{z_{id}\}$ and parameters θ_d and ϕ_k. [10 points]

   (c) Integrate out the parameters $\boldsymbol{\theta}_d$'s and $\boldsymbol{\phi}_k$'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}, and B_{kw}. [5 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 the following identity of the Gamma function: $\Gamma(1 + x) = x\Gamma(x)$ for $x > 0$. [10 points]

   (e) What hyperpriors would you give to α and β. Propose and derive a sampling update for α and β? [5 points]

2. **[65 points] Decrypting Messages with MCMC**. You are given a passage of English text that has been encrypted by remapping each symbol to a (usually) different one. For example,

$$\begin{array}{rcl} a & \rightarrow & s \\ b & \rightarrow & ! \\ \langle\text{space}\rangle & \rightarrow & v \\ \ldots & \ldots & \ldots \end{array}$$

Thus a text like '`a boy`...' might be encrypted by '`sv!op`...'. Assume that the mapping between symbols is one-to-one. The file **symbols.txt** gives the list of symbols, one per line (note second line is ⟨space⟩). The file **message.txt** gives the encrypted message.

Decoding the message by brute force is impossible, since there are 53 symbols and thus 53! possible permutations to try. Instead we will set up a Metropolis-Hastings Markov chain to find modes in the space of permutations.

We model English text, say $s_1 s_2 \cdots s_n$ where s_i are symbols, as a Markov chain, where each symbol given the immediately previous one is independent of all earlier symbols:

$$p(s_1 s_2 \cdots s_n) = p(s_1) \prod_{i=2}^{n} p(s_i | s_{i-1})$$

(a) Learn the transition statistics of the English language. In particular, download a large text, say *War and Peace* (in translation!), from the web and estimate symbol probabilities $p(s) = \phi(s)$ and transition probabilities $p(s|t) = \psi(s,t)$. You may ignore the initial symbol probabilities in the following.

Give formulas for the ML estimate of these probabilities as functions of the counts of numbers of occurrences of symbols and pairs of symbols.

Compute and report athese estimated probabilities in a table. [6 marks]

(b) The state variable for our MCMC sampler will be the permutation amongst the symbols. Let $\sigma(s)$ be the symbol that stands for symbol s in the encrypted text, e.g. $\sigma(a) = s$ and $\sigma(b) =!$ above. Assume a uniform prior distribution over permutations.

Are the latent variables $\sigma(s)$ for different symbols s independent?

Let $e_1 e_2 \cdots e_n$ be an encrypted English text. Write down the joint probability of $e_1 e_2 \cdots e_n$ and $s_1 s_2 \cdots s_n$ given σ. [6 marks]

(c) We shall use a Metropolis-Hastings (MH) sampler, with a proposal formed by choosing two symbols s and s' at random and swapping the corresponding encrypted symbols $\sigma(s)$ and $\sigma(s')$.

What is the probability of a given proposal, and what is the corresponding acceptance probability? [10 marks]

(d) Implement the MH sampler, and run it on the included encrypted text. Report the current decryption of the first 60 symbols after every 100 iterations. Your Markov chain should hopefully converge to give you a fairly sensible message. (Hint: it may help to initialize your chain intelligently and to try multiple times; if any case, please describe what you did). [30 marks]

(e) Note that some $\psi(s,t)$ values may be zero. Does this affect the ergodicity of the chain? If the chain remains ergodic, give a proof; if not, explain and describe how you can restore ergodicity. [5 marks]

(f) Analyse this approach to decoding. For instance, would symbol probabilities alone (rather than pairwise stats) be sufficient? If we used a second order Markov chain for English text, what problems might we encounter? Will it work if the encryption scheme allows two symbols to be mapped to the same encrypted value? Would it work for Chinese with > 10000 symbols? [8 marks]

3. **[Bonus 60 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. [20 points]

   (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. [5 points]

   (d) Try varying α, β and K. What effects do these have on the posterior and predictive performance of the model? Justify your answers. [5 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.

   (e) 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]

   (f) 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 α, β and K until the topics discovered looks "reasonable". Describe the topics you found. How do the topics change (qualitatively) as α, β and K are varied? [10 points]