# Assignment 4

## Probabilistic and Unsupervised Learning

Yee Whye Teh & Maneesh Sahani

Due: Thurs Dec 4, 2008

**Note:** all assignments for this course are to be handed in to the Gatsby Unit, **not** to the CS department. Please hand in all assignments at the beginning of lecture on the due date to the lecturer. Late assignments will be penalised. If you are unable to come to class, you can also hand in assignments to Rachel Howes in the Alexandra House 4th floor reception.

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 LDA is a topic model—multiple mixture models with shared components—with the following conditional probabilities:

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

   Assume our data consists of $D$ documents, a vocabulary of size $W$, and we 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 $k$.

   5% 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 $B_{kw}$.

   10% Derive the Gibbs sampling updates for all the latent variables and parameters.

   10% Integrate out the parameters $\boldsymbol{\theta}_d$'s and $\boldsymbol{\phi}_k$'s from the joint probability above, 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 $B_{kw}$.

   10% 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 $x > 0$.

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, e.g.,

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

Thus a text like "a boy..." might get encrypted by "sv!op...". Assume that each symbol is mapped to a unique symbol. 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 thus 53! permutations. Instead you will set up a Metropolis-Hastings Markov chain to find modes on the space of permutations.

We model English text, say $s_1 s_2 s_3 \cdots s_n$ where $s_i$ are symbols (a,...,z,<space>,.,;,...), as a Markov chain, where a symbol given the previous symbol is independent of all past symbols:

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

Learn some statistics of the English language. In particular, download a large text, say *War and Peace*, 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.

2% Give formulas for the ML estimate of these probabilities as functions of counts of numbers of occurrences of symbols and pairs of symbols.

5% Report these estimated probabilities in a table.

The state of the system consist of the permutation among the symbols. Let $\sigma(s)$ be the symbol which symbol $s$ is encrypted as, e.g. $\sigma(\mathsf{a}) = \mathsf{s}$ and $\sigma(\mathsf{b}) = !$ above. We assume a uniform prior distribution over permutations.

3% Are the latent variables $\sigma(s)$ for symbols $s$ independent?

5% Let $e_1 e_2 \cdots e_n$ be an encrypted English text. Write down the probability of $e_1 e_2 \cdots e_n$ given $\sigma$.

10% We shall use a Metropolis-Hastings (MH) sampler. The proposal is to pick two symbols and swap the symbols with which these two symbols are encrypted with. What are the proposal distributions and acceptance probabilities?

30% 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; if you do, please describe what you did).

10% Discuss variations on the decoder. For instance: will symbol probabilities alone be sufficient? if we use a second order Markov chain for English text, what problems might we encounter? Will it work if two symbols can be mapped to the same symbol? Will it work for Chinese with $> 10000$ symbols?

3. [**Bonus 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.

$4 \times 5\%$ 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% 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% 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% Try varying $\alpha$, $\beta$ and $K$. What effects do these have on the posterior and predictive performance of the model? Justify your answers.

**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`.

10% 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% 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?