STA 250 :: Homework Policy

For all questions you must show your work. This enables us to understand your thought process, give partial credit and prevent crude cheating. Please see the code of the conduct in the Syllabus for rules about collaborating on homeworks.

For questions requiring computing, if you use `R`, `python` *or any programming environment then you must turn in a printout of your output with your solutions.*
In addition, a copy of your code must be uploaded to the appropriate `HW` *directory of your forked GitHub repo within 24 hours of the homework due date.*

Homework 01

Due: In Class, Mon October 28th

Assigned: Wednesday Oct 16th

Bayesian Inference Module

In this homework you will fit a Bayesian model (or models) using MCMC, validate your code using a simulation study, and then apply your method to a real dataset.

0. Sync your fork of the course GitHub repo to include the latest updates using the instructions provided here.

1. In this question we consider some basic proofs of the correctness of the Gibbs sampling algorithm.

    a. Suppose we have a two-dimensional target density $p(x_1, x_2)$. Using the characterization of the stationary distribution presented in class, prove that the stationary distribution of the two-component Gibbs sampler is the desired target distribution $p(x_1, x_2)$.
    Notes:
        - You may assume that the Markov chain generated by the Gibbs sampler is ergodic.
        - The Gibbs sampler is a *very* commonly used algorithm. There are lots of proofs of its convergence on/off-line. Please do not use a proof from another source: prove it yourself!

    b. Again, assuming that the resulting Markov chain is ergodic, prove that the stationary distribution of the p-component Gibbs sampler is indeed $p(x_1, x_2, \ldots, x_p)$.

2. We begin by considering the logistic regression model:

$$y_i|\beta \sim \text{Bin}(m_i, \text{logit}^{-1}(x_i^T\beta)), \qquad i = 1, \ldots, n.$$

where $\beta \in \mathbb{R}^p$,

$$\text{logit}^{-1}(u) = \frac{\exp(u)}{1+\exp(u)},$$

and the y_i's are conditionally independent given β. We will place a simple multivariate normal prior on β i.e.,

$$\beta \sim N(\mu_0, \Sigma_0).$$

a. Write down the posterior distribution for β up to proportionality.

b. Navigate to the `HW1/BayesLogit` directory of your GitHub repo on Gauss. Run the script `BLR_sim.sh` on Gauss by executing the command:

```
sarray ./BLR_sim.sh
```

After the job completes, you should have 200 data files, and 200 parameter files inside the data folder. The datasets are generated with the following specifications:

- $p = 2$: An intercept, and one covariate per observation
- $\mu = (0, 0)^T$: Zero prior mean for β
- $\Sigma = \text{diag}(1, 1)$: Diagonal unit prior covariance matrix for β

c. Copy one of the data files (any will do), and its corresponding true parameter values, from Gauss to your laptop/desktop.

d. Implement an MCMC routine `blr_fit.[R,py]` to fit the Bayesian Logistic Regression model to the dataset you have copied to your laptop/desktop. If using `R` you should write a function with prototype:

```
"bayes.logreg" <- function(m,y,X,beta.0,Sigma.0.inv,
                    niter=10000,burnin=1000,
                    print.every=1000,retune=100,
                    verbose=TRUE){
    ...
}
```

If using `Python` use:

```
def bayes_logreg(m,y,X,beta_0,Sigma_0_inv,
```

```
niter=10000,burnin=1000,
print_every=1000,retune=100,
verbose=True):
```

In both cases the arguments are as follows:

- `m`: Vector containing the number of trials for each observation (of length n)
- `y`: Vector containing the number of successes for each observation (of length n)
- `X`: Design matrix (of dimension $n \times p$)
- `beta.0`: Prior mean for β (of length p)
- `Sigma.0.inv`: Prior precision (inverse covariance) matrix for β (of dimension $p \times p$)
- `niter`: Number of iterations to run the MCMC after the burnin period
- `burnin`: Number of iterations for the burnin period (draws will not be saved)
- `print.every`: Print an update to the user after every period of this many iterations
- `retune`: Retune the proposal parameters every `return` iterations. No tuning should be done after the burnin period is completed
- `verbose`: If `TRUE` then print lots of debugging output, else be silent

Run the code on the dataset, and verify that it produces posterior credible intervals that cover the true parameter values (at least to some extent).

The script must output the $1, 2, \ldots, 99\%$ percentiles of the marginal posterior distributions for β_0 and β_1 to file in a 99 row and 2 column `.csv` (with no header).

e. Now that you have written an MCMC function to fit a single dataset, copy your working code file `BLR_fit.[R,py]` to the `BayesLogit` directory of your repo on Gauss. Modify the code to take a single command line argument (an integer from 1 to 200) and analyze the corresponding dataset from the `results` directory. Please see the files `BLR_fit_skeleton.R` and/or `BLR_fit_skeleton.py` for further details.

f. Run `BLR_fit_R.sh` or `BLR_fit_py.sh` to fit the 200 simulated datasets. These are array job scripts so run using:

```
sarray BLR_fit_[R,py].sh
```

After analyzing each dataset, you should have 200 results files in the `results` directory:

```
$ ls results/
blr_res_1001.csv  blr_res_1002.csv  ...  blr_res_1200.csv
```

Again, each file should contain 99 rows and 2 columns corresponding to the $1, 2, \ldots, 99\%$ percentiles of the marginal posterior distributions of β_0 and β_1 i.e.,

```
$ head results/blr_res_1001.csv -n 2
-0.615697082535097,0.594913779947118
-0.610734788568127,0.601946320177968
```

g. Run the script `blr_post_process.R` on Gauss using:

```
sbatch blr_post_process.sh
```

The result will be a file `coverage_line_plot.pdf` showing the empirical vs. nominal coverage for your analysis of the 200 datasets. Numerical results will also be stored in `coverage_summaries.txt` and `coverage_summaries.tex`. If your coverage lines stray outside the guiding bands then you may want to check your code for bugs or run your MCMC for more iterations. (The guiding bands are based on a simple pointwise approximate Binomial calculation, so it is possible that your lines could fall outside the bands, but if they do so consistently then you likely have a problem!).

If your coverage plot passes the test then congratulations -- you have coded your Bayesian Logistic Regression MCMC algorithm successfully.

What to turn in:

i. The analytic form of the posterior distribution, up to proportionality.
ii. A description of your MCMC algorithm. This should address the following questions:

- What proposal distribution did you use?
- What was your tuning process (if you had one)?
- How many iterations did you run your sampler for?
- How long was your burnin period?
- How did you choose your starting values?

iii. The numerical coverage properties in `coverage_summaries.txt` or `coverage_summaries.tex` (placed inside a full report).
iv. The graph of the coverage properties in `coverage_line_plot.pdf`

3. Next up, time to analyze a real dataset using your MCMC code. The dataset `breast_cancer.txt` in the GitHub repo is taken from the UCI machine learning repository, see here for full references. The dataset has 10 covariates, and a single response variable (either "M" for malignant, or "B" for benign).

a. Fit a Bayesian logistic regression model to the breast cancer dataset, using all covariates, and an intercept term (for the response: set "M" to be 1, and "B" to be zero). Use the following prior specifications: $\beta = (0, \ldots, 0)^T$ and $\Sigma = \text{diag}(1000, \ldots, 1000)$.
b. Compute the lag-1 autocorrelation for each component of β.
c. Which covariates seem to be related to the cancer diagnosis?
d. Perform a posterior predictive check, using the statistic (or statistics) of your choice.
e. Is your model a reasonable fit to the data or not? Discuss.

(: Happy Coding! :)