CS 6190: Probabilistic Modeling **Due Tu 4/12**

Homework 3: MCMC

Instructions: Be sure to electronically submit your answers in either R Markdown (*.Rmd) or Sweave (*.Rnw) format. Include all of the output of your code, plots, and discussion of the results in your write up. You may work together and discuss the problems with your classmates, but write up your final answers entirely on your own.

1. Write a function to perform Gibbs sampling of a binary label image x with the Ising model prior and iid Gaussian likelihood, given a noisy image y. This function should take α, β, and σ parameters, and generate a random binary image (labels in the set $\{-1, 1\}$) according to the posterior Gibbs distribution for $x \mid y$. The energy should look like this:

$$U(x) = -\alpha \sum_i x_i - \beta \sum_{\langle i,j \rangle} x_i x_j + \frac{1}{2\sigma^2} \sum_i (x_i - y_i)^2.$$

   Note this is assuming that the x_i labels are also the mean pixel values in the Gaussian. The α parameter controls the proportion of labels that are -1 versus $+1$. Negative values of α will favor more -1 pixels, and positive values of α will favor more $+1$ pixels. You can use functions for reading and displaying PNG images provided in the file `MRF.r`.

   (a) Run your code with just the Ising prior term (no posterior). Do this a few times and generate several random binary images. Try different α and β terms to see what the effects are.

   (b) Run your code on the images `noisy-message.png` and `noisy-yinyang.png`. Compute posterior mean images for both examples (again, don't forget to burn-in). You can fix values for α, β, and σ^2 (you will want to tune these manually to get something that works well). What values of α, β, and σ^2 did you use?

   (c) Use your posterior samples to iteratively estimate σ^2 from the data. That is, assume the "clean" image that you sample is the true image, and use it to get an MLE of σ^2 (update this estimate each iteration). What final estimate do you get for σ^2?

2. Say you are given data (X, Y), with $X \in \mathbb{R}^d$ and $Y \in \{0, 1\}$. The goal is to train a classifier that will predict an unknown class label $\tilde{y}$ from a new data point $\tilde{x}$. Consider the following model:

$$Y \sim \text{Ber}\left(\frac{1}{1 + e^{-X^T \beta}}\right),$$
$$\beta \sim \text{N}(0, \sigma^2 I).$$

   This is a **Bayesian logistic regression** model. Your goal is to derive and implement a Hamiltonian Monte Carlo sampler for doing Bayesian inference on β.

   (a) Write down the formula for the unormalized posterior of $\beta \mid Y$, i.e.,

$$p(\beta \mid y; x, \sigma) \propto \prod_{i=1}^{n} p(y_i \mid \beta; x_i) p(\beta; \sigma)$$

(b) Show that this posterior is proportional to $\exp(-U(\beta))$, where

$$U(\beta) = \sum_{i=1}^{n}(1 - y_i)x_i^T\beta + \log(1 + e^{-x_i^T\beta}) + \frac{1}{2\sigma^2}\|\beta\|^2.$$

(c) Implement a Hamiltonian Monte Carlo routine in R for sampling from the posterior of β. Feel free to use the code from the Radford Neal paper.

(d) Use your code to analyze the `iris` data in R, looking only at two species, *versicolor* and *virginica*. The species labels are your Y data, and the four features, petal length and width, sepal length and width, are your X data. Also, add a constant term, i.e., a column of 1's to your X matrix. Use the first 30 rows for each species as training data and leave out the last 20 rows for each species as test data (for a total of 60 training and 40 testing). Generate samples of β (don't forget to burn-in), and use these to get a prediction, $\tilde{y}$, of the class labels for the test data. Use your samples to get a Monte Carlo estimate of the posterior predictive probability $p(\tilde{y}|y)$ for each testing data point.

(e) Draw trace plots of your β sequence and histograms (do 1D plots of each four vector components separately).

(f) Compare this to the true class labels, y, and see how well you did by estimating the average error rate, $\mathrm{E}[|y - \tilde{y}|]$ (a.k.a. the zero-one loss). What values of σ, ϵ, and L did you use?

Optional Extensions: If you want to go beyond what's required, try extending your logistic regression to a multinomial logistic regression (i.e., more than two categories). Another possible extension would be to use a kernel (e.g., squared exponential) on the x values.