CSC 2541, Spring 2011 — Major Assignment

Due at the start of class on March 21. Worth 25% of the course mark.

This assignment is to be done by each student individually. You may discuss it in general terms with other students, but the work you hand in should be your own. In particular, you should not leave any discussion with someone else with any written notes (either paper or electronic).

In this assignment, you will implement a Gaussian process regression model in which some observations are "outliers", with larger noise variance than ordinary observations.

The data consists of n observations, $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i is the vector of predictor variables for observation i, and y_i is the real-valued response in this observation.

Which observations are outliers is modeled in terms of unobserved binary variables $o_1, \ldots, o_n$, with $o_i = 1$ if observation i is an outlier. The full model is as follows, with the only dependencies being those that follow from those shown explicitly:

$$\begin{aligned}
y_i \,|\, x_i,\, o_i,\, f,\, \sigma_0,\, \sigma_1 &\sim \begin{cases} N(f(x_i),\, \sigma_0^2) & \text{if } o_i = 0 \\ N(f(x_i),\, \sigma_0^2 + \sigma_1^2) & \text{if } o_i = 1 \end{cases} \\
f \,|\, \gamma,\, \rho &\sim GP(\gamma, \rho) \\
o_i \,|\, \theta &\sim \text{Bernoulli}(\theta) \\
\theta &\sim \text{Beta}(1, 19) \\
\log(\gamma) &\sim N(0, 3^2) \\
\log(\rho) &\sim N(0, 3^2) \\
\log(\sigma_0) &\sim N(-1, 3^2) \\
\log(\sigma_1) &\sim N(1, 3^2)
\end{aligned}$$

The Gaussian process $GP(\gamma, \rho)$ has mean zero and covariance function

$$K(x, x') \;=\; \text{Cov}(f(x), f(x')) \;=\; 10^2 \,+\, \gamma^2 \exp\Big(-\,\rho^2\, ||x - x'||^2\Big)$$

Since the function f is infinitely complex, you can't represent it explicitly in your program. Instead, you should integrate it away to give the multivariate Gaussian joint density of the responses conditional on the predictors, outlier indicators, and parameters: $P(y_1, \ldots, y_n \,|\, x_1, \ldots, x_n, o_1, \ldots, o_n, \gamma, \rho, \sigma_0, \sigma_1)$. The θ parameter can also be easily integrated away.

You should use MCMC to implement Bayesian inference for the remaining variables and parameters, log transforming the parameters so they will be unconstrained (and probably otherwise easier to update too). The state of the Markov chain will therefore consist of $o_1, \ldots, o_n$, $\log(\gamma)$, $\log(\rho)$, $\log(\sigma_0)$, $\log(\sigma_1)$. You should first try what's probably the most obvious MCMC method, in which you update each of these in turn, using univariate Metropolis updates. You will need to find suitable proposal distributions for the Metropolis updates of the parameters. Initially, the proposal distribution you use for the o_i should be to propose the value opposite the current value (ie, if $o_i = 0$, propose $o_i = 1$; if $o_i = 1$, propose $o_i = 0$).

You should then see if you can improve the efficient of this MCMC method by changing the proposal distribution for the o_i. The proposal distribution can be non-symmetric, in which case you will need to use Metropolis-Hastings updates, in which the ratio of proposal probabilities affects the acceptance probability (see the last two slides of the Week 4 lecture). The general idea, which you should implement in some suitable form, is to look at the "residuals" for the observations, given by $y_i - p_i$, where p_i is the posterior mean value for a response in another observation with the same predictor variables, x_i. These residuals will be conditional on the current values of all the o_i and of the model parameters. It seems plausible that if the absolute value of the residual is large compared to σ_0, the outlier indicator o_i is likely to be 1, and conversely, if the residual is small, o_i is likely to be 0. This suggest using a proposal distribution for o_i that with fairly high probability (but not probability one) proposes the value that is "more likely" by some such criterion.

Note again that, as mentioned above, for non-symmetric proposal distributions, the acceptance probability has to account for the lack of symmetry, which requires computing the probability of proposing the current state from the proposed state. Note that this backwards probability might well be found using different residuals than the forward probability.

The computational saving with this scheme will occur when the value proposed in this way happens to be the same as the current value. This proposal can obviously be accepted (with no effect) without any further computation. When a value different from the current value is proposed, costly matrix operations will be needed to decide whether to accept it. These costly operations are needed for every update with the original scheme. On the other hand, it's conceivable that the old scheme might move more rapidly around the posterior distribution. So it's not obvious which is better.

I have provided a synthetic data set with $p = 5$ predictors on the course web site, as a file with one line for each observation, containing the five predictor values followed by the response value, with spaces separating them. The file has 1100 observations. You are to take the first $n = 100$ of these observations as the training set. The remaining 1000 observations can be used as a test set. This assignment does not require assessing performance of predictions for these test cases, but you may be curious, and they may be useful for debugging.

You should start by implementing the basic MCMC scheme described first above, adjusting the proposal distributions for the parameters to get reasonable sampling performance. You should then implement some version of the second scheme (you'll have to fill in the details), and compare its sampling efficiency with that of the first scheme. You should avoid inefficiencies of an algorithmic nature (such as re-computing something you expensively computed before and could have saved), but there's no need to spend much time on detailed coding optimizations.

Sampling inefficiency can be measured by the product of average computing time per iteration (here, an update of all the variables in turn) and the autocorrelation time of some suitable function of state (such as one of the paramerters, or the sum of all the o_i). See the last few slides from Week 3 for a discussion of autocorrelation time.

You should hand in your program, a discussion of how you implemented the MCMC methods, a discussion of the results of comparing them, and any plots or other output that support your conclusions.