Rare event estimation: Quiz 3

Email answers & code to Patrick Laub before 5pm on March 29

1. These questions consider the toy problem from the lectures $\ell = \mathbb{P}(Z > 5)$ for $Z \sim \mathsf{Normal}(0, 1)$.

   (a) Write some code, or simply copy it from the lectures, which samples from $g^*(x) = 1\{x > 5\} f_Z(x)/\ell$ using MCMC with transition kernel

   $$q(x_r \mid x_{r-1}) = \frac{1}{2\lambda} \exp\left\{ -\frac{|x_r - x_{r-1}|}{\lambda} \right\}.$$

   In other words, use a *random walk sampler* where jumps are $\mathsf{Laplace}(\lambda)$-sized.

   (b) The MCMC algorithm is an approximate algorithm, and samples generated using it are not as valuable as exact i.i.d. samples generated using other methods (like acceptance-rejection). To quantify this difference, one can calculate the *effect sample size* (ESS) as

   $$\text{ESS} = \frac{R}{1 + 2\sum_{i=1}^{\infty} \rho(i)}$$

   where $\rho(i)$ is the (auto-)correlation between each sample X_r and the sample X_{r-i}) which lags behind it by i.

   Run MCMC multiple times where each run uses a different value of λ using a grid so that $0 < \lambda$ and $\lambda \leq 10$. Calculate the ESS for each λ, and report the λ^* which corresponds to the best λ (the one which gives the largest ESS).

   Some R code which implements the ESS calculation is attached in the appendix below; feel free to use this.

   (c) Calculate the fraction of proposals which were accepted using this best λ^*.

   Hint: the `diff` function evaluated on a vector $\{X_1, \ldots, X_R\}$ returns $\{X_2 - X_1, X_3 - X_2, \ldots, X_R - X_{R-1}\}$, and if the i-th proposal is rejected we'll have $X_i = X_{i-1}$.

2. These questions consider $\ell = \mathbb{P}(M > 3)$ where $M = \max\{X_1, X_2\}$ for $\boldsymbol{X} = (X_1, X_2) \sim \mathsf{Normal}(\mathbf{0}, \mathbf{\Sigma}_M)$ where $\mathbf{\Sigma}_M = [1, 0.8; 0.8, 1]$. This is the same problem from Quiz 1.

   (a) Write a function which takes a number $\sigma^2 > 0$ and returns an MCMC sample from the bivariate distribution

   $$g^*(x_1, x_2) = \frac{1\{\max\{x_1, x_2\} > 3\} f_{\mathbf{0_2}, \mathbf{\Sigma}_M}(x_1, x_2)}{\ell}$$

   where $f_{\boldsymbol{\mu}, \mathbf{\Sigma}}(\boldsymbol{x})$ denotes the p.d.f. of the d-dimensional (here $d = 2$) normal distribution with mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$,

   $$f_{\boldsymbol{\mu}, \mathbf{\Sigma}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d \det(\mathbf{\Sigma})}} \exp\Big\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^2 \mathbf{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})^2\Big\}.$$

   Use the transition kernel $q(\mathbf{y} \mid \mathbf{x}) = f_{\mathbf{x}, \sigma^2 \mathbf{I}}(\mathbf{y})$. In other words, use a *random walk sampler*, where proposal points are generated by adding $\mathsf{Normal}([0, 0], [\sigma^2, 0; 0, \sigma^2])$-sized jumps to the current point.

   Start the MCMC at $\mathbf{X}^0 = (3.1, 3.1)$, and generate $10^3 + 10^5$ points. Discard the first 10^3 points, this is our *burn in period*. Have the function return the remaining 10^5 points.

   (b) Run *improved cross entropy* (ICE) to estimate ℓ by searching inside the

   $$f(\boldsymbol{x}; v) = \frac{1}{\sqrt{2 \det(\mathbf{\Sigma}_M)}} \exp\Big\{-\frac{1}{2}(\boldsymbol{x} - v\mathbf{1}_2)^2 \mathbf{\Sigma}_M^{-1} (\boldsymbol{x} - v\mathbf{1}_2)^2\Big\}$$

   family for the optimal parameter v^*. Note, this is the 'improved' version of the Question 4 from Quiz 1.

   (c) Compare the ICE v^* solution against your (or my) equivalent result using the original cross-entropy method from Quiz 1. If they differ significantly, try using different values of σ^2 in the MCMC sampling above to get the two algorithms to return roughly the same value.

A Effective sample size code

Adapted from `https://github.com/tpapp/MCMCDiagnostics.jl/blob/master/src/MCMCDiagnostics.jl`:

```
autocorrelation <- function(x, k, v) {
   R <- length(x)
   x1 <- (x[1:(R-k)]); x2 <- (x[(1+k):R])
   V <- sum((x1 - x2)^2) / length(x1)
   return(1 - V / (2*v))
}

ess_estimate <- function(x)  {
   v <- var(x); R <- length(x)
   tau_inv <- 1 + 2 * autocorrelation(x, 1, v)
   K <- 2
   while (K < R - 2) {
      delta <- autocorrelation(x, K, v) +
               autocorrelation(x, K + 1, v)

      if (delta < 0) {
         break
      }

      tau_inv <- tau_inv + 2*delta
      K <- K + 2
   }

   return( R * min(1 / tau_inv, 1) )
}
```