STAT 447 Assignment 9

MCMC Hacking

Caden Hewlett

2024-04-04

Data Import

We import the primary data set inspired by Davidson-Pilon, (2013) below:

```r
# main data vector
sms_data = c(
    13,24,8,24,7,35,14,11,15,11,22,22,11,57,11,19,29,6,19,
    12,22,12,18,72,32,9,7,13,19,23,27,20,6,17,13,10,14,6,
    16,15,7,2,15,15,19,70,49,7,53,22,21,31,19,11,18,20,12,
    35,17,23,17,4,2,31,30,13,27,0,39,37,5,14,13,22)
# data frame
df = data.frame(
  num_texts = sms_data,
  day = 1:length(sms_data)
)
```

Bayesian Model

We denote C to be the change point, selected uniformly from days $d \in \{1, 2, \dots N\}$ where N is the number of observations. Then, there is a likelihood for days less than change point C and a likelihood for days above the change point.

We can denote the model as follows:

$$\begin{aligned} \lambda_1 &\sim \exp(1/100) \\ \lambda_2 &\sim \exp(1/100) \\ C &\sim \text{unif}\big(\{1, 2, \dots, N\}\big) \\ Y_d \mid C, \{\lambda_1, \lambda_2\} &\sim \text{pois}\big(\mathbb{1}[\,d < C\,]\lambda_1 + \mathbb{1}[\,d \geq C\,]\lambda_2\big) \end{aligned}$$

We will also refer to the $\{\lambda_1, \lambda_2\}$ pair as $\vec{\lambda}$.

We provide an implementation of the joint distribution of this model below.

```r
# inputs are lambdas, C and y
log_joint = function(rates, change_point, y) {

  # Return log(0.0) if parameters are outside of the support
  if (rates[[1]] < 0 | rates[[2]] < 0 | change_point < 1 | change_point > length(y))
    return(-Inf)
```

```r
  log_prior =
    dexp(rates[[1]], 1/100, log = TRUE) +
    dexp(rates[[2]], 1/100, log = TRUE)

  log_likelihood = 0.0
  for (i in 1:length(y)) {
    rate = if (i < change_point) rates[[1]] else rates[[2]]
    log_likelihood = log_likelihood + dpois(y[[i]], rate, log = TRUE)
  }

  return(log_prior + log_likelihood)
}
```

Question 1: A Custom MCMC Sampler

NOTE We will briefly justify the reasoning for irreducibility of each kernel (and thus the combination of kernels) in Part 1 and leave invariance for the next section. Further, when we define these kernels, we will be doing so in terms of likelihoods γ. However, for more precision with respect to the true implementation, we will prove π-invariance with respect to the log joint model defined above.

Part 1: Algorithm

We will begin by defining two separate kernels K_1 and K_2 for the rate parameters $\{\lambda_1, \lambda_2\}$ and the change-point parameter C, respectively. We then unify these kernels by defining a kernel mixture for a selection probability ρ.

Let $C^\star$ and C be the proposed and current cutoff times, respectively. Similarly, we let $\vec{\lambda}^\star$ and $\vec{\lambda}$ be the proposed and current rates, respectively. We use $\vec{\lambda} = \{\lambda_1, \lambda_2\}$ and $\vec{\lambda}^\star = \{\lambda_1^\star, \lambda_2^\star\}$ interchangeably. Further, in each case, we define the proposal function using the following notation:

$$q\Big(\{\vec{\lambda}^\star, C^\star\} \mid \{\vec{\lambda}, C\}\Big) \equiv q\Big(\{\lambda_1^\star, \lambda_2^\star, C^\star\} \mid \{\lambda_1, \lambda_2, C\}\Big)$$

While slightly obtuse, this is intended to reflect the fact that the `rates` input is a vector, and the likelihood defined in the original model relies upon all three parameters. Further, this style allows us to maintain the same general form for q regardless of the kernel, which we can use to expedite the proof in Question 2.

We begin by discussing K_1. This kernel will only modify the $\vec{\lambda}$ parameter. Since each Poisson rate parameter is a real number, we will use a standard normal proposal.

Notably, rather than having a dimension of 2, we have each rate operate separately on their own means to be updated over time. The objective is to have more independent exploration for pre-and-post change point rates. Notably, our proposal vector is $\{\vec{\lambda}^\star, C\}$, highlighting that in this kernel we do not select new values for the change point.

$$q\Big(\{\vec{\lambda}^\star, C\} \mid \{\vec{\lambda}, C\}\Big) = \Big\{\frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{1}{2}(\lambda_i^\star - \lambda_i)^2\Big) : i \in \{1, 2\}\Big\}$$

$$\alpha\big(\vec{\lambda}^\star, \vec{\lambda}, C\big) = \min\Big\{1, r\big(\vec{\lambda}^\star, \vec{\lambda}, C\big)\Big\}, \text{ where } r\big(\vec{\lambda}^\star, \vec{\lambda}, C\big) = \frac{\gamma(\lambda^\star, C)}{\gamma(\lambda, C)}$$

$$K_1\big(\vec{\lambda}^\star \mid \vec{\lambda}\big) = q\big(\{\vec{\lambda}^\star, C\} \mid \{\vec{\lambda}, C\}\big) \cdot \alpha\big(\vec{\lambda}^\star, \vec{\lambda}, C\big)$$

In a simplified version, we can write the equation below to simplify K_1 :

$$K_1\big(\vec{\lambda}^\star \mid \vec{\lambda}\big) = q\big(\vec{\lambda}^\star \mid \vec{\lambda}\big)\alpha\big(\vec{\lambda}^\star, \vec{\lambda}\big)$$

Now, we consider K_C. Ideally, we would like to define a symmetric proposal q_C; however, we cannot use a Normal Distribution as before so we must find a symmetric discrete distribution. As per the recommendation, this distribution should have variance greater than 3 to avoid slow mixing.

The simplest possible proposal in this situation would be the discrete $\text{unif}\big(\{a = 1, b = N\}\big)$ distribution, i.e.:

$$q_C(x' \mid x) = \frac{1}{b - a + 1} = \frac{1}{N - 1 + 1} = \frac{1}{N}$$

Where, in this case, $N = 74$. This choice ensures that every possible change point in the support of C can be proposed with nonzero probability.

We can verify by computing the variance of this distribution (to check it will mix nicely):

$$\text{var}\Big(\text{unif}\,\{a, b\} \mid a = 1, b = N\Big) = \frac{(b - a + 1)^2 - 1}{12} = \frac{(N - 1 + 1)^2 - 1}{12} = \frac{N^2 - 1}{12} \approx 456 \text{ for } N = 74$$

Which is certainly a large enough variance to have a breadth of proposal options.

Then, as before, we define the acceptance probability $\alpha_C(x' \mid x)$ as:

$$\alpha_C(x' \mid x) = \min\big\{1, r(x', x)\big\} = \min\left\{1, \frac{\gamma(x')}{\gamma(x)}\right\}$$

We also know that this element of the kernel is irreducible, assuming $\pi(x), \pi(x') > 0$. We know that this is the case because for any arbitrary x', there is a finite number of steps m with nonzero probability required to reach the destination state. In other words, $\mathbb{P}(X^{(m)} = x' | X^{(0)} = x) > 0$. We know that this is the case because the m-step transition kernel $K_m = \mathbb{P}(X^{(m)} = x' | X^{(0)} = x) > 0$ is nonzero for all valid x, x' under the restrictions defined earlier.

Let's outline this fact using an arbitrary $K_m(x \mid x')$:

$$\begin{aligned}
K_m(x \mid x') &= \mathbb{P}(X^{(m+1)} = x' | X^{(m)} = x) \\
K_m(x \mid x') &= q_C(x' \mid x) \times \alpha_C(x' \mid x) \\
K_m(x \mid x') &= \frac{1}{N} \times \min\left\{1, \frac{\gamma(x')}{\gamma(x)}\right\} \\
K_m(x \mid x') &= \frac{1}{N} \times \min\left\{1, \frac{p_{\text{pois}}(x'; \lambda)}{p_{\text{pois}}(x; \lambda)}\right\}
\end{aligned}$$

Where the rate parameter λ is simplified from $\mathbb{1}[\,d < C\,]\lambda_1 + \mathbb{1}[\,d \geq C\,]\lambda_2 \equiv$ Let's take a moment to verify the term in the MH Ratio. Ideally, $r(x, x')$ should be well-defined and nonzero $\forall (x, x')$ that could be proposed from $q_C(x' \mid x)$. We assume that x is already well-proposed (i.e. $x^{(0)}$ isn't bad). Directly, we know by definition of the Poisson PMF that $\forall x \in \mathbb{Z}^+, p_X(x) > 0$. Hence, the ratio is well defined.

Since the ratio is well defined, we know by the above that it will always be the case that $K_m(x \mid x') > 0$, hence all steps are accessible from one another. Thus, K_C does not "break" the irreducibility clause in the MCMC algorithm.

Part 2: π-Invariance

Prove that the MCMC algorithm you defined in part 1 is π-invariant.

Since detailed balance implies global balance, it is sufficient to show that the DBEs hold to show invariance. We know that if each K_i in the kernel mixture is π-invariant that their mixture is π-invariant. Hence, we will separately prove π-invariance for K_λ and K_C hold under detailed balance to show that K is invariant.

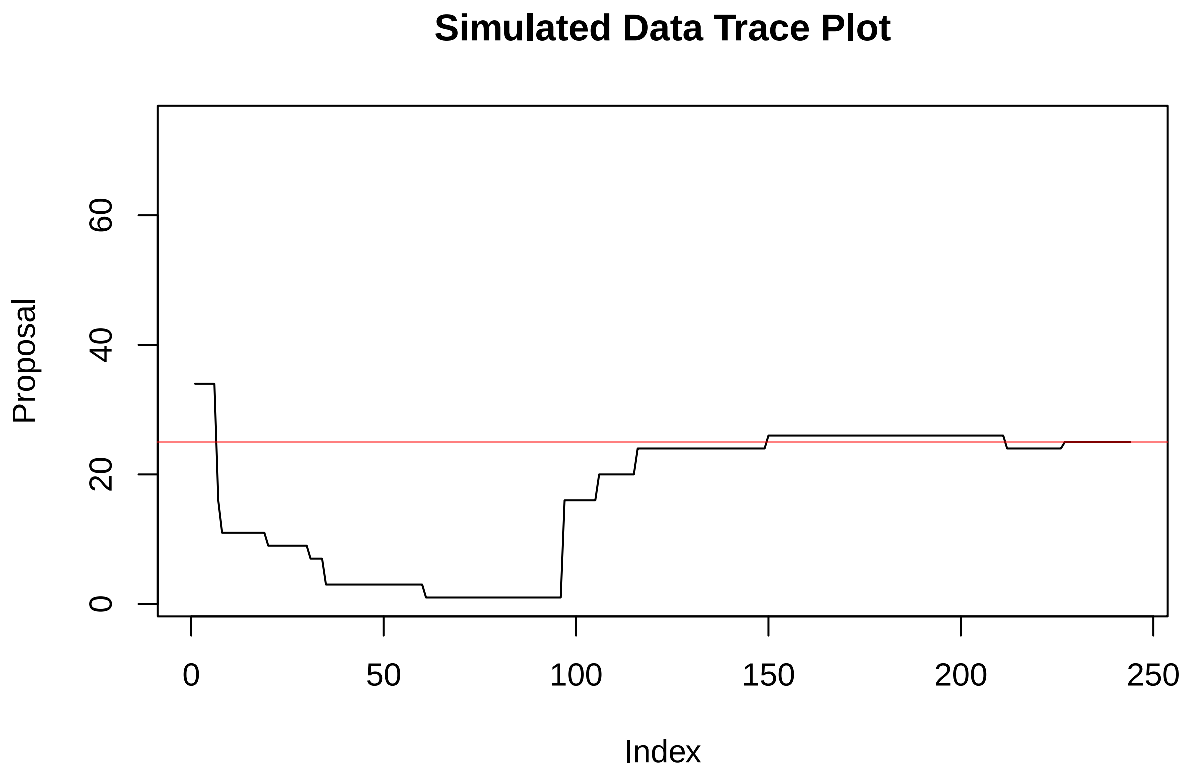
Part 3: Implementation

Implement the MCMC algorithm you describe mathematically in `R`.

```r
N = nrow(df)
# we build such that dim = 1
mcmc = function(rates, CP, y, n_iterations, debug = FALSE) {
  change_point_trace = rep(-1, n_iterations)
  # initial point
  current_CP = CP
  current_RT = rates
  current = list(rates, CP)
  # iteration station
  for (i in 1:n_iterations) {
    if (debug) {print(unlist(rep("#", times = 10)))}
    if (debug) {print(paste("Iteration:", i))}
    # bernoulli trial
    kernel_choice = ifelse(runif(1) < 0.5, 1, 2)
    if (debug) {print(paste("Chose Kernel", kernel_choice, "..."))}
    # Kernel for Change Point
    if (kernel_choice == 1){
      # Discrete Uniform Proposal
      prop = rdunif(1, 1, N)
      if (debug) {print(paste("Proposed", prop, "..."))}
      # MH ratio
      ratio = (log_joint(current[[1]], prop, y) -
                log_joint(current[[1]], current[[2]], y))
      # Bernoulli Trial
      if (log(runif(1)) < ratio) {
        # accept
        current[[2]] = prop
        if (debug) {print("Accepted!")}
      } else {
        # reject (redundant but nice)
        current[[2]] = current[[2]]
        if (debug) {print("Rejected!")}
      }
    }
    # Kernel for Lambdas
    else {
      # normal at current point
      ell_1 = rnorm(1, mean = current[[1]][1])
      ell_2 = rnorm(1, mean = current[[1]][2])
      # then the proposal is the vector
      prop = c(ell_1, ell_2)
      if (debug) {print(paste("Proposed", unlist(round(prop,2)), "..."))}
      # MH Ratio
      ratio = (log_joint(prop, current[[2]], y) -
                log_joint(current[[1]], current[[2]], y))
      if (log(runif(1)) < ratio) {
        # accept
        current[[1]] = prop
        if (debug) {print("Accepted!")}
      }else {
```

```r
        # reject (redundant but nice)
        current[[1]] = current[[1]]
        if (debug) {print("Rejected!")}
      }
    # update trace
     change_point_trace[i] = current[[2]]
     if(debug) {Sys.sleep(3)}
    }
  }
  return(
    list(
      change_point_trace = change_point_trace,
      last_iteration_rates = current[[1]]
    )
  )
}
```

```r
## TESTING
set.seed(447)
# true change point at around 25
simulated_yvals = c(round(runif(24, 10, 15)), round(runif(60, 30, 50)))
# run MCMC
test = mcmc(c(0.1, 0.2), 34, simulated_yvals, 550, debug = FALSE)
# plot
plot(main = "Simulated Data Trace Plot",
     test$change_point_trace[test$change_point_trace > 0],
     type = 'l', ylab = "Proposal",
     ylim = c(1, 74))
abline(h = 25, col = rgb(1, 0, 0, 0.5), lwd = 1)
```



Simulated Data Trace Plot
Proposal
0
20
40
60
0
50
100
150
200
250
Index