Computational statistics
Homework 3

Salomé Do

February 8, 2019

Exercise 7.22. : Travelling salesman problem with Metropolis-Hastings Algorithm

The travelling salesman problem (TSP) can be formulated as follows : a salesmans has to visit $N \in \mathbb{N}$ customers, all living in different places $\{1, ..., N\}$. The salesmans minds the planet, so he doesn't want to spend too much gas visiting his customers : he wants to drive as few miles as possible. Thus, he must choose the shortest route to visit his N customers. We simulate [1] this problem with $N = 30$ places randomly determined.

Figure 1: Places to visit in a simulation of the TSP

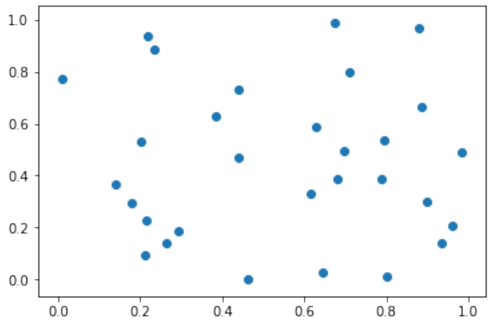
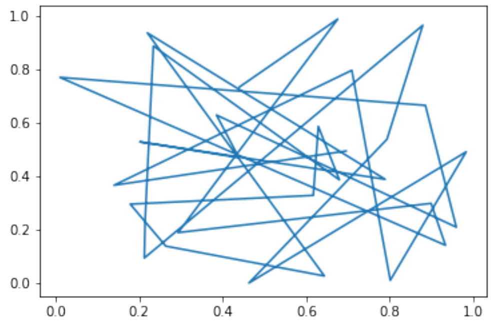


Figure 2: A random route to visit the places (d=16.4)



The problem can be expressed as follows, considering d a distance on all the places $1, ..., N$, and the permutations $\sigma \in \mathcal{S}^N$ which define the order in which the travelling salesman sees his customers:

$$\min_{\sigma \in \mathcal{S}^N} \sum_i d(i, \sigma(i))$$

The TSP problem can be tackled through a Metropolis Hastings algorithm. Let's define:

$$H(\sigma) = \sum_i d(i, \sigma(i))$$

[1]The code can be found here

H is the objective function to minimize, with regard to a permutation σ.

Following a Simulated-Annealing scheme, we want to :
At step i, given a permutation σ_i:

1. Simulate ζ a new permutation candidate, from an instrumental density $g(|\zeta - \sigma_i|)$
2. Compute:
$$\rho_i = min\{\exp(\frac{\Delta h_i}{T_i}), 1\}$$
and take $\sigma_{i+1} = \zeta$ with probability ρ_i, $\sigma_{i+1} = \sigma_i$ otherwise.
3. Update T_i to T_{i+1}

To implement this algorithm, two things are to be precised : the instrumental density g, and the 'temperature' function T_t. First, we represent a permutation $\sigma \in \mathcal{S}^N$ as a vector of dimension N, in the form of :
$$\sigma = \begin{pmatrix} \sigma(1) \\ \sigma(2) \\ \vdots \\ \sigma(N) \end{pmatrix}$$
At each step i, we want to randomly generate a candidate permutation ζ, close to σ_i. An idea for that is to randomly chose a pair of coordinates in the σ_i vector and permute them. This randomly generates a new permutation ζ close to σ_i. Regarding the temperature function T_t, we can choose, as a first guess, the example from section 5.2.3, namely $T_t = \frac{1}{log(t)}$.

We then run the algorithm on our toy example generated in Figure 1, with 100000 simulations.

Figure 3: Route after 100 000 iterations

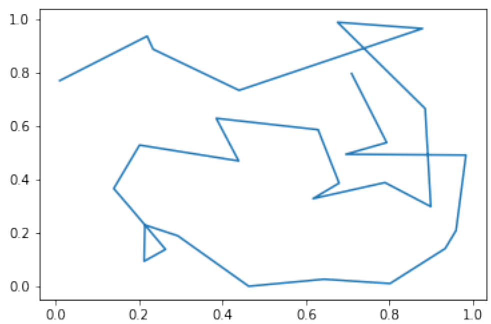
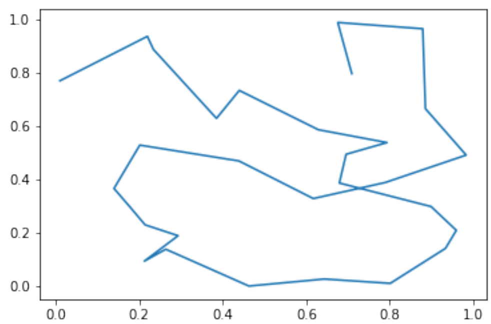
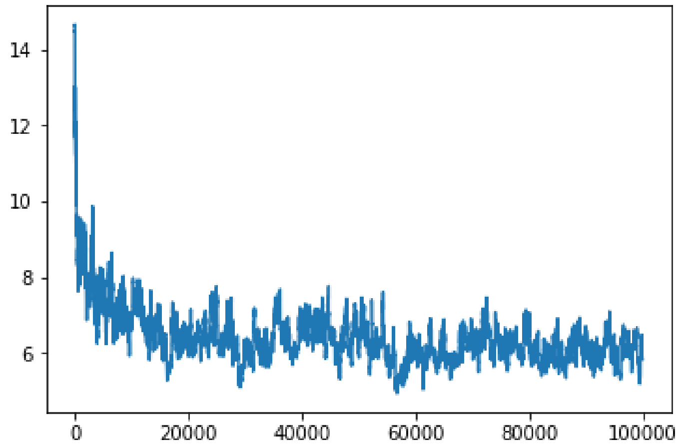


Figure 4: Shortest route amongst all (d=4.9)



The first thing to note is that the last route isn't the shortest route amongst the simulations. Figure 5, which plots each simulation's distances, gives an idea of the convergence of the algorithm. In our case, the total distance of the route is quickly reduced, but the convergence seems to be pretty poor.

Figure 5: Total distance for each route generated by the algorithm



As this might be due to the temperature function, we chose to explore other temperature functions. We first tried functions of the form $T_t = \frac{1}{\log(t^a)}, a \in \mathbb{N}$. Although a acts just as a regularization constant (since $\frac{1}{\log(t^a)} = \frac{1}{a\log(t)}$), the parameter changes the appearance of the convergence plot. We tested value from 1 to 100 for a, and averaged the results of 20 runs of the algorithm per a value. The algorithm ran 1000 Simulated-Annealing iterations per simulation. The value giving the shortest route in average was $a = 86$, and the shortest distance was $d = 6.22$ (Figure 7) amongst 20 runs for this a value, thus a worse result than before. However, the convergence for $a = 86$ (Figure 6) was clearer.

Figure 6: Convergence for $a = 86$

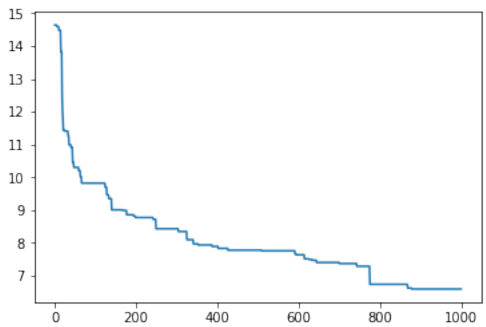
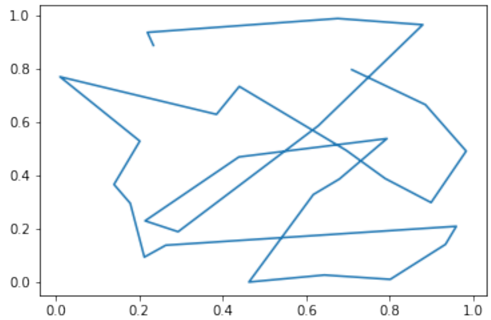


Figure 7: Best route for all 20 runs in $a = 86$



We gave another try to another form of temperature function : $T_t = \frac{1}{t^2}$. Running 100000 simulations gives the following convergence scheme (Figure 8), which is satisfying. The shortest route has a total distance of 5.33 (Figure 9), which is near the results of the first temperature function $T_t = \frac{1}{\log(t)}$. The $T_t = \frac{1}{t^2}$ temperature function has thus the advantage of giving good results, while showing less variability than the previous functions.

Figure 8: Convergence scheme

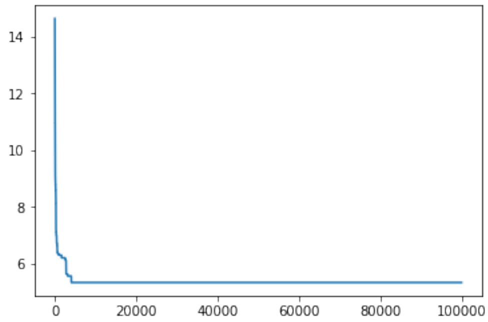
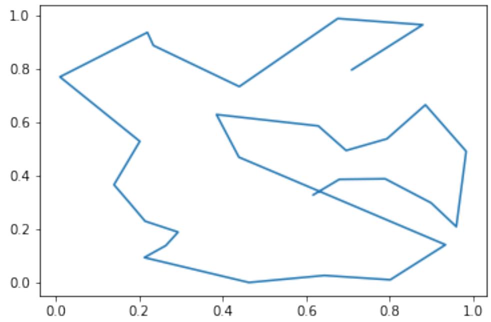


Figure 9: Shortest route (d=5.33)



Exercise 10.10 Multi-stage Gibbs Sampler on Clinical Mastisis Data

Overview of the problem

Let $X_i, i = 1, ..., m$ be the number of cases of mastisis (an inflammation caused by infection) in herd i. Assuming that the occurrence or not of a mastisis for a single animal is a Bernouilli random variable, a model for the herd could be: $X_i \sim \mathcal{P}(\lambda_i)$, where λ_i is the infection rate. The main problem here is the independence which could lead to larger parameter estimates variances. Schukken et al. 1991 use thus a hierarchical model :

$$\begin{aligned} X_i &\sim \mathcal{P}(\lambda_i) \\ \lambda_i &\sim \mathcal{G}a(\alpha, \beta_i) \\ \beta_i &\sim \mathcal{G}a(a, b) \end{aligned}$$

The objective is to estimate λ_i by simulation, using the Gibbs sampler.

Gibbs Sampler

In the Multi-stage Gibbs Sampler context, we want to simulate $X = (X_1, ..., X_p)$ and we suppose that we are able to simulate from univariate conditionnal densities $f_1, ..., f_p$, so that :

$$X_i | x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_p \sim f_i(x_i | x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_p)$$

The Gibbs Sampler simulates x^{t+1} from x^t by doing the following updates at time t :

Generate :

- $X_1^{t+1} \sim f_1(x_1 | x_2^t, ..., x_p^t)$
- $X_2^{t+1} \sim f_2(x_2 | x_1^{t+1}, x_3^t ..., x_p^t)$
- ...
- $X_p^{t+1} \sim f_p(x_p | x_1^{t+1}, ..., x_{p-1}^{t+1})$

In our case, λ_i, β_i are to be simulated. Let's first compute their posterior densities.

(a) Posterior densities computation

We know that $\lambda_i \sim \mathcal{G}a(\alpha, \beta_i)$. Thus,

$$\pi(\lambda_i = \lambda \mid \alpha, \beta_i) = \lambda^{\alpha-1}\frac{\beta_i^\alpha e^{-\beta_i\lambda}}{\Gamma(\alpha)}, \forall \lambda \in \mathbb{R}^+$$

Additionally : $f(x_i|\lambda_i = \lambda, \alpha, \beta_i) = \frac{\lambda^{x_i}}{x_i!}e^{-\lambda}$. Then, from Bayes Theorem :

$$\pi(\lambda_i = \lambda \mid x, \alpha, \beta_i) = \frac{f(x_i|\lambda_i = \lambda, \alpha, \beta_i)\pi(\lambda_i = \lambda| \alpha, \beta_i)}{\underbrace{\int_0^\infty f(x_i|\lambda_i = s, \alpha, \beta_i)\pi(\lambda_i = s| \alpha, \beta_i)ds}_{A}}$$

. Let's first compute the integral A.

$$\begin{aligned}
A &= \int_0^\infty \frac{s^{x_i+\alpha-1} \times e^{-(\beta_i+1)s} \times \beta_i^\alpha}{x_i! \times \Gamma(\alpha)}ds \\
&= \frac{\beta_i^\alpha}{x_i! \times \Gamma(\alpha)}\int_0^\infty s^{x_i+\alpha-1} \times e^{-(\beta_i+1)s}ds \\
&= \frac{\beta_i^\alpha}{x_i! \times \Gamma(\alpha) \times \beta_i^{x_i+\alpha}}\int_0^\infty u^{x_i+\alpha-1}e^{-u}du & \leftarrow u := (\beta_i+1)s \\
&= \frac{\beta_i^\alpha}{x_i! \times \Gamma(\alpha) \times \beta_i^{x_i+\alpha}} \times \Gamma(x_i+\alpha)
\end{aligned}$$

Then,

$$\begin{aligned}
\pi(\lambda_i = \lambda \mid x, \alpha, \beta_i) &= \frac{x_i! \times \Gamma(\alpha) \times \beta_i^{x_i+\alpha}}{\beta_i^\alpha \times \Gamma(x_i+\alpha)}\lambda^{\alpha-1}\frac{\beta_i^\alpha e^{-\beta_i\lambda}}{\Gamma(\alpha)}\frac{\lambda^{x_i}}{x_i!}e^{-\lambda} \\
&= \frac{\lambda^{x_i+\alpha-1}(\beta_i+1)^{x_i+\alpha}e^{-(\beta_i+1)\lambda}}{\Gamma(x_i+\alpha)}
\end{aligned}$$

Which is the density of a $\mathcal{G}a(x_i+\alpha, \beta_i+1)$.
Now, show that $\pi(\beta_i|x, \alpha, a, b, \lambda_i) = \mathcal{G}a(\alpha+a, \lambda_i+b)$. Denote, for clarity:

$$f(\beta) = \pi(x_i|\lambda_i, \beta_i = \beta, \alpha, a, b)\pi(\lambda_i|\beta_i = \beta, \alpha, a, b)\pi(\beta_i = \beta|a, b), \forall \beta \in \mathbb{R}^+$$

From Bayes Theorem, for any $\beta \in \mathbb{R}^+$,

$$\pi(\beta_i = \beta|x, \alpha, a, b, \lambda_i) = \frac{f(\beta)}{\int_0^\infty f(s)ds}$$

Using the respective laws of X_i, λ_i, β_i :

$$\begin{aligned}
f(\beta) &= \frac{\lambda_i^{x_i}}{x_i!}e^{-\lambda_i}\lambda_i^{\alpha-1}\frac{\beta^\alpha e^{-\beta\lambda_i}}{\Gamma(\alpha)}\beta^{a-1}\frac{b^a e^{-b\beta}}{\Gamma(a)} \\
&= \underbrace{\frac{\lambda_i^{x_i+\alpha-1}e^{-\lambda_i}b^a}{x_i!\Gamma(\alpha)\Gamma(a)}}_{C}\beta^{\alpha+a-1}e^{-\beta(\lambda_i+b)}
\end{aligned}$$

With C a constant regarding β. Thus, the integral value can be calculated as follows :

$$\begin{aligned}
\int_0^\infty f(s)ds &= \int_0^\infty C s^{\alpha+a-1} e^{-s(\lambda_i+b)} ds \\
&= C \int_0^\infty \frac{1}{(\lambda_i+b)^{\alpha+a}} t^{(\alpha+a)-1} e^{-t} dt \qquad \leftarrow t = s(\lambda_i+b) \\
&= \frac{C}{(\lambda_i+b)^{\alpha+a}} \Gamma(\alpha+a)
\end{aligned}$$

Thus,

$$\begin{aligned}
\pi(\beta_i = \beta | x, \alpha, a, b, \lambda_i) &= \frac{C\beta^{\alpha+a-1} e^{-\beta(\lambda_i+b)} (\lambda_i+b)^{\alpha+a}}{C\Gamma(\alpha+a)} \\
&= \beta^{\alpha+a-1} \frac{(\lambda_i+b)^{\alpha+a} e^{-\beta(\lambda_i+b)}}{\Gamma(\alpha+a)}
\end{aligned}$$

Which is the density of a $\mathcal{G}a(\alpha+a, \lambda_i+b)$.

(b) Gibbs sampler implementation

We implemented [2]

[2] Code available here